Radio Shack®

TRS-80 Model PC-4

Pocket Computer

Programming Guide

Introduction to Computing With TRS-80 Pocket Computer Model PC-4

- Step-by-Step Instructions
- Programs for Statistics, Games, Finance And More!

Cat. No. 26-3650

PREFACE

Today personal computers can be seen wherever you go. They are in wide use as tools for both business and research. As a result, software is becoming more and more important.

BASIC, a language used for personal computers, can be understood by everyone, because it uses words found in everyday conversation.

BASIC is the first step in making various kinds of work more efficient. Consequently, more and more people in various fields are becoming interested and BASIC is gaining wider recognition.

But at the same time, there seems to be an increasing number of people who shy away from BASIC because they think it is difficult to learn.

When it comes to BASIC programming, as in many other things, "practice makes perfect". However, many people who would like to learn it are having difficulty finding the time and place to do so.

This manual and the PC-4 have been designed for use at almost any time or location. The manual is written in a clear, easy-to-follow format. We feel that you will enjoy learning from it while actually practicing various programming techniques with the PC-4 Personal Computer. Short programs are used to provide ample opportunity for improving your skill.

This manual places emphasis on gaining practical experience. That's why in the very first chapter the question "What can I use the PC-4 for?" is asked and answered.

Even without knowing the grammar, you can input short programs and try to execute them. BASIC grammar is written in easily understandable terms that anyone, given some practice, can comprehend. Interesting examples have been included in the program library. These have been developed with the assistance

of noted authorities in the computer field.

By making your own modifications, you will be able to find out just what can be done using BASIC, all the while becoming more and more skillful.

By using this manual, soon every member of the family will become a BASIC fan.

CONTENTS

Chapter 4 *Now Let's Learn BASIC.*

Chapter 5 *Program Library*

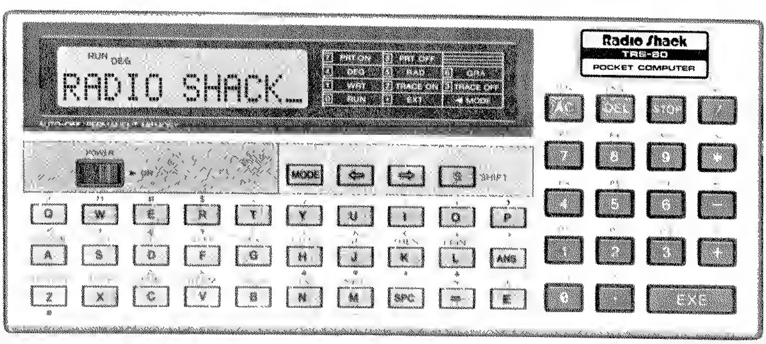
Chapter 1

What Is the PC-4 Used for ?

1.1 Learning BASIC by using the PC-4

The computer has now advanced to a level where it is used not only by experts but also by grammar school students.
As a result of the progress made in semi-conductor technology, the computer has entered an age when it can be made smaller and smaller.
At first glance, the PC-4 looks like a calculator with a lot more keys.
Personal computers, which can be placed on a table, are talked about as being the standard of the microcomputer age.
On the other hand, the PC-4 is a "pocketable computer". This does not mean however that the PC-4 is just an oversized calculator. On the contrary, it is a genuine computer in every respect. This is because the PC-4 uses a high-level programming language known as BASIC and that can be used to perform a wide range of tasks.
BASIC is a language which is used in the small business field or in technological applications. It is not a difficult language to learn. Even grammar school students can use it after some training.
A high-level language means that we can program using words which are similar to those used in daily conversation, rather than very special computer codes. Of course, you cannot become a programming expert with only two or three days of study. However, if you try the program examples in this manual, one by one, you will naturally gain a better understanding of BASIC.
As will be stated over and over in this manual, when it

comes to programming, doing is more important than reading. So follow the examples and press the keys. An "ERR" (ERROR) display may appear often, but, if you try to figure out the cause, you will slowly gain an understanding of the various operations.
This manual has been written so that you can refer to specific pages for guidance whenever you have difficulty. So let's relax and learn BASIC.



1.2 What do we use the PC-4 for ?

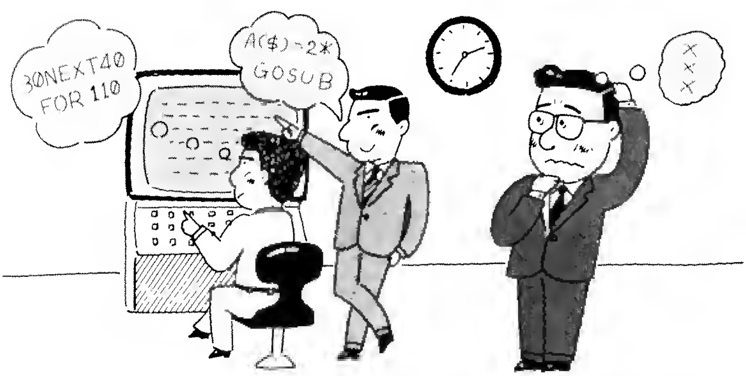
"PC-4 Application Map"

1.3 I want to master BASIC!

The PC-4 is a splendid device for beginning the study of BASIC. One reason for this is that the BASIC which is used in the PC-4 is very close to standard BASIC (the internal circuitry is standard).

The following program is a short but typical example of BASIC. Almost all personal computers can run it.

This program includes a fundamental loop and a substitution statement. Using the BASIC learned with the PC-4 you can take on larger computers when you have to.



For those who would like to master BASIC, input the program shown below using the operating procedure. This is a program to obtain the total of numbers from 1 through N For example, if 10 is input for N, the result will be 55.

Program to obtain the total of numbers from 1 through N

```
P0
 10 INPUT "N= ",N
 20 S=0
 30 FOR I=1 TO N
 40 S=S+I
 50 NEXT I
 60 PRINT S
 70 GOTO 10
```

[Explanation]
This Program contains fundamental BASIC commands. If you can understand this program, it means that you have mastered the fundamentals of BASIC. The meanings of the commands and symbols which are used here are explained in Chapters 2 through 4. After trying this program out by inputting various numbers for N, refer to Chapter 2 for a more through explanation.

[Program input procedure]

Power ON

‹1› [MODE] [1] [C] [L] [E] [A] [R] [A] [EXE]

The display will read as follows.

Means that a program can be written. (Omitted hereafter)

```
 WRT DEG
P 0 1 2 3 4 5 6 7 8 9
```

Blinks (Ready for writing in P0)

‹2› [1] [0] [S] [M] (INPUT) [S] [W] (") [N] [=]
[SPC] [S] [W] (") [S] [P] (,) [N] [EXE]

Display at this time

```
 10 INPUT " N=
```

☞ [S] *is the Shift Key. Do not confuse it with the regular letter* [S].

< 3 > [2] [0] [S] [=] [0] [EXE]

Confirm the display. Does it read as follows?

```
20    S=0
```

< 4 > [3] [0] [S] [FOR S] [I] [=] [1] [S] [TO D] [N] [EXE]
< 5 > [4] [0] [S] [=] [S] [+] [I] [EXE]
< 6 > [5] [0] [S] [NEXT G] [I] [EXE]
< 7 > [6] [0] [S] [PRINT L] [S] [EXE]
< 8 > [7] [0] [S] [GOTO H] [1] [0] [EXE]

This completes the input. You should have confirmed the display for operations < 4 > through < 8 > as well. If a mistake is made during the above operation, press [⇦] and [⇨] to move the blinking dash (cursor) to the position you want to correct and reinput the correct information.

[Program execution procedure]

(1) After the [EXE] Key is pressed for operation < 8 > above, press [S] [P0 0] and the display will be as follows.

```
P _1 2 3 4 5 6 7 8 9
```

(2) [MODE] [0]

(3) [S] [Po / Ø] The display will be as follows.

```
N= ?
```

(4) Input a number for N.
For example, if N=10, input the following.

[1] [Ø] [EXE]

(5) Then the result will be displayed.

```
5 5
```

If the [EXE] Key is pressed again, the program will repeat from the beginning.

```
N= ?
```

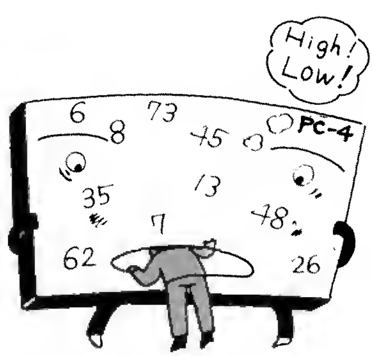
1.4 I want to play a game !

There are probably a lot of people who would like to use the PC-4 to play games. There is no reason why you can't enjoy yourself and learn BASIC at the same time. With the PC-4, there is no capability to have large objects jumping around on the screen because the display is rather small. Nevertheless, there are numerous mental games which can be enjoyed using the PC-4. Several of these are contained in this manual.
If programs are recorded on a cassette tape using your tape recorder, you can enjoy various games at any time by calling out the programs from the tape. Well, let's play.

HI-LO Game

[Explanation]

At the beginning of the game, the PC-4 hides a number. The object of the game is for you to guess what the number might be. When you venture some number using the numerical keys, the PC-4 will display whether the number you guessed is higher than ("HI : K=") or lower than ("LO:K=") the hidden number. By repeating this cycle, you can get closer and closer to the hidden number. How good is your sixth sense?



For those people, who say "I want to play a game.", input this program into the PC-4 by using the following procedure. This game is called the "number guessing game" or the "HI-LOW game".
See how many turns it takes to guess the number. This game can be enjoyed by two or more people.

HI-LO Game program

```
P0
  10 PRINT "HI-LO"
  20 K=0:INPUT "KEY
     IN LENGTH ",L
  30 R=INT (RAN#*10↑
     L)
  40 INPUT "KEY IN N
     O.",A:K=K+1:H$=
     "HI"
  50 IF A=R THEN 80
  60 IF A<R;H$="LO"
  70 PRINT H$;":K=";
     K:GOTO 40
  80 PRINT "GOOD:ANS
     =";R:PRINT "K="
     ;K:GOTO 20
```

[Program input procedure]

Power ON

<1> [MODE] [1] [C] [L] [E] [A] [R] [A] [EXE]

Display at this time

```
 WRT  DEG  Omitted hereafter
P 0 1 2 3 4 5 6 7 8 9
  ↑
  Blinks
```

<2> [1] [0] [S] [L] (PRINT) [S] [W] (") [H] [I] [−] [L] [O] [S] [W] (") [EXE]

Display at this time

```
10   PRINT   "HI
```

〈3〉 [2] [0] [K] [=] [0] [S] [I] [S] INPUT [M]
[S] [W] [K] [E] [Y] [SPC] [I] [N] [SPC]
[L] [E] [N] [G] [T] [H] [SPC] [S] [W] [S] [P]
[L] [EXE]

Display at this time

```
20  K  =  0:INPUT
```

Prior to pressing the [EXE] Key for the last time, corrections can be made by moving the cursor to the desired position using the direction keys ([◁] [▷]) and inputting the correct information.

〈4〉 [3] [0] [R] [=] [I] [N] [T] [S]
[T] [R] [A] [N] [S] [E] [*] [1]
[0] [S] [·] [L] [S] [Y] [EXE]

〈5〉 [4] [0] [S] INPUT [M] [S] [W] [K] [E]
[Y] [SPC] [I] [N] [SPC] [N] [O]
[·] [S] [W] [S] [P] [A] [S] [I]
[K] [=] [K] [+] [1] [S] [I] [H] [S] [R] [=]
[S] [W] [H] [I] [S] [W] [EXE]

〈6〉 [5] [0] [S] IF [J] [A] [=] [R] [S]
THEN [K] [8] [0] [EXE]

〈7〉 [6] [0] [S] IF [J] [A] [S] [/] [R] [S] [O]
[H] [S] [R] [=] [S] [W] [L] [O] [S] [W]
[EXE]

〈8〉 [7] [0] [S] PRINT [L] [H] [S] [R] [S] [O] [S]
[W] [S] [I] [K] [=] [S] [W] [S] [O] [K]
[S] [I] [S] GOTO [H] [4] [0] [EXE]

〈9〉 [8] [0] [S] PRINT [L] [S] [W] [G] [O] [O] [D]
[S] [I] [A] [N] [S] [=] [S] [W]

[S] [O] [R] [S] [I] [S] [L] [S]
[W] [K] [=] [S] [W] [S] [O] [K]
[S] [I] [S] [H] [2] [0] [EXE]

This completes the input.

[Program execution procedure]

(1) After pressing the [EXE] Key for <9> above, press [S] [0] and the display will be as follows.

```
P_ 1 2 3 4 5 6 7 8 9
```

(2) [MODE] [0] The display will be as follows.

```
RUN DEG
R E A D Y   P 0
```

(3) Press [S] [0]. Does the display show HI-LO? If so, press the [EXE] Key and KEY IN LENGTH? will be displayed.

This is asking you "How many digits do you want in the number to be guessed?". You can designate a number such as 1 through 10; if you press [2] [EXE] , the hidden number will be 0 through 99, and if you press [3] [EXE] , the number will be 0 through 999. Here, press [2] [EXE] , then KEY IN NO? is displayed and the game begins. If a number from 0 through 99 is input followed by [EXE], the display will read LO:K=1 or HI:K=1. LO means too low and HI means too high. K is the number of guesses.

Since KEY IN NO.? is displayed again by pressing [EXE] , input an appropriate number. By repeating this, you can get closer and closer to the hidden number. When you succeed, GOOD:ANS=00 (hidden number) is displayed, and K=00 (number of trials) is given by pressing [EXE] .

1.5 I want to use it for accounting or business applications !

You probably realize that computers are used in today's society for performing a variety of functions. The outstanding feature of a computer is its ability to handle and process many calculations. An endless number of business applications, such as sales accounting, stock control, calculation of loan payments, making estimates and cost accounting, can be performed more efficiently with the aid of a computer.

The PC-4 can perform all of these functions with few limitations.

We will use the calculation of loan payments as an example. With it, you can determine the proper periodic payment and surprise the salesman.

Calculation of monthly loan payment

Suppose you are buying a new automobile and the sales price is $10,000 with an annual interest rate of 11% and 24 monthly payments. In this case, what is the monthly payment when the down payment was $1,000, $2,000, etc.?

The calculation is performed using the following formula.

$$P = (PT - R) \times \frac{\frac{i}{12}}{1 - \frac{1}{\left(1 + \frac{i}{12}\right)^n}}$$

P: Monthly payment
PT: Price
R: Down payment
n: Number of payments
i: Interest (annual rate)

For P, the payment is rounded up to the nearest dollar.

There are probably a lot of people who will say, "I want to use the PC-4 for accounting or business applications." Monthly loan payment calculations such as this, where a certain formula is used, can be done using a scientific calculator. However, using BASIC, anyone can obtain the result quickly and easily.

Program for monthly loan payment

```
P0
  10 INPUT "PRICE",A
  20 INPUT "ANNUAL I
     NTEREST(%)",I
  30 I=I/1200
  40 INPUT "NUMBER O
     F PAYMENTS",N
  50 INPUT "DOWN PAY
     MENT",R
  60 X=(A-R)*I
  70 Y=1-1/(1+I)↑N
  80 K=INT(X/Y+.99)
  90 PRINT "MONTHLY
     PAYMENT=";K
 100 GOTO 50
```

[Explanation]

The calculation can be made with such a short program. The X on line 6Ø is the numerator and the Y on line 7Ø is the denominator. X/Y is calculated on line 8Ø and the result is rounded up to the nearest dollar. The PC-4 will request the price, annual interest, number of payments and down payment. All you have to do is input the numbers.

[Program input proceduure]

‹1› [MODE] [1] [C] [L] [E] [A] [R] [A] [EXE] [S] [V] (DEFM) [0] [EXE]

Display will show ***VAR : 26

‹2› [1] [0] [S] [M] (INPUT) [S] [W] (") [P] [R] [I] [C] [E] [S] [W] (") [S] [P] (,) [A] [EXE]

‹3› [2] [0] [S] [M] (INPUT) [S] [W] (") [A] [N] [N] [U] [A] [L] [SPC] [I] [N] [T] [E] [R] [E] [S] [T] [S] [T] (() [MODE] [·] [S] [Q] (%) [MODE] [·] [S] [Y] ()) [S] [W] (") [S] [P] (,) [I] [EXE]

‹4› [3] [0] [I] [=] [I] [/] [1] [2]

[0] [0] [EXE]

‹5› [4] [0] [S] [M] [S] [W] [N] [U] [M] [B] [E] [R] [SPC] [O] [F] [SPC] [P] [A] [Y] [M] [E] [N] [T] [S] [S] [W] [S] [P] [N] [EXE]

‹6› [5] [0] [S] [M] [S] [W] [D] [O] [W] [N] [SPC] [P] [A] [Y] [M] [E] [N] [T] [S] [W] [S] [P] [R] [EXE]

‹7› [6] [0] [X] [=] [S] [T] [A] [−] [R] [S] [Y] [*] [I] [EXE]

‹8› [7] [0] [Y] [=] [1] [−] [1] [/] [S] [T] [1] [+] [I] [S] [Y] [S] [.] [N] [EXE]

‹9› [8] [0] [K] [=] [I] [N] [T] [S] [T] [X] [/] [Y] [+] [·] [9] [9] [S] [Y] [EXE]

‹10› [9] [0] [S] [L] [S] [W] [M] [O] [N] [T] [H] [L] [Y] [SPC] [P] [A] [Y] [M] [E] [N] [T] [=] [S] [W] [S] [O] [K] [EXE]

‹11› [1] [0] [0] [S] [H] [5] [0] [EXE]

This completes program input.

[Program execution procedure]

(1) After pressing [EXE] at step < 11 > above, press [MODE] [0] [S] [0] and the display will be as follows.

PRICE ?

(2) Input 10000 and press [EXE] and the display will be as follows.

ANNUAL INTEREST(%)?

(3) Hereafter, input the numbers following the display.
Input 11 and press [EXE] and the display will be as follows.

NUMBER OF PAYMENTS ?

(4) Input 24 and press [EXE] and the display will be as follows.

DOWN PAYMENT ?

(5) Input 1ØØØ and press [EXE] and the result will be as follows.

MONTHLY PAYMENT = 42Ø.

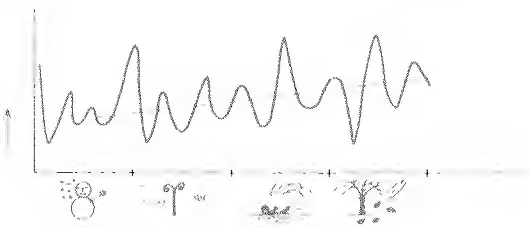
(6) If you press [EXE] , the program will return to Step <4>. This time, input 2ØØØ and press [EXE] and the result, $373, will be displayed. This can be repeated using various amounts.

1.6 I want to use it for statistics or prediction calculation !

The library section of this manual has examples of deviation value calculations and statistical calculations with two variables. These apply to the totaling and analysis of scores, to data analysis for quality control, and to prediction based on sampling.

When you would like to know, for instance, the trend of a value which has a certain tendency, you can find it out using the moving average. For example, the stock prices on the stock market are changing daily. However, if you look at them at weekly or monthly intervals, you will see that they are going up or coming down. If you obtain the moving average, it is possible to predict the timing with which the stock market rises and falls.

An example of prediction based on sampling is shown below.



Product sales and production planning

For many products, there is a seasonal variation. So let's predict the proper production quantity for six months from now using the moving average value for past sales statistics. The past statistics are as follows.

Month	Apr.	May	June	July	Aug.	Sep.	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.
Sales quantity of product A	549	248	411	857	192	235	480	331	928	459	366	529
Moving average	—	—	—	—	451.4	388.6	435	419	433.2	486.6	512.8	522.6

For those who wonder, "What can be done using statistical or prediction calculations?", try the following example. A 5-month moving average was obtained from the past year's data. Using an 8-month moving average, let's make a chart to predict 6 months into the future.

Program to obtain the moving average

```
P0    10 VAC
      20 INPUT "NUMBER O
         F MOVES=",A
      30 B=B+1
      40 PRINT "DATA";B+
         A*C
      50 INPUT F(B)
      60 IF C=0;IF B<A T
         HEN 30
      70 D=0
      80 FOR E=1 TO A
      90 D=D+F(E)
     100 NEXT E
     110 PRINT "AVERAGE=
         ";D/A
     120 IF B=A;B=0:C=C+
         1
     130 GOTO 30
```

[Program input procedure]

〈1〉 [MODE] [1] [C] [L] [E] [A] [R] [A] [EXE] [S] [V] (DEFM) [2] [1] [EXE]

Display at this time.

```
***VAR:47
```

(Data input memory has been designated.)

〈2〉 [1] [0] [V] [A] [C] [EXE]

〈3〉 [2] [0] [S] [M] (INPUT) [S] [W] (") [N] [U] [M] [B] [E] [R] [SPC] [O] [F] [SPC] [M] [O] [V] [E] [S] [=] [S] [W] (") [S] [P] (,) [A] [EXE]

〈4〉 [3] [0] [B] [=] [B] [+] [1] [EXE]

〈5〉 [4] [0] [S] [L] (PRINT) [S] [W] (") [D] [A] [T] [A] [S] [W] (") [S] [O] (;) [B] [+] [A] [*] [C] [S] [O] (;) [EXE]

〈6〉 [5] [0] [S] [M] (INPUT) [F] [S] [T] (() [B] [S] [Y] ()) [EXE]

〈7〉 [6] [0] [S] [J] (IF) [C] [=] [0] [S] [O] (;) [S] [J] (IF) [B] [S] [Z] (<) [A] [S] [K] (THEN) [3] [0] [EXE]

〈8〉 [7] [0] [D] [=] [0] [EXE]

<9> [8] [0] [S] [S] (FOR) [E] [=] [1] [S] [D] (TO) [A] [EXE]

<10> [9] [0] [D] [=] [D] [+] [F] [S] [T] (() [E] [S] [Y] ()) [EXE]

<11> [1] [0] [0] [S] [G] (NEXT) [E] [EXE]

<12> [1] [1] [0] [S] [L] (PRINT) [S] [W] (") [A] [V] [E] [R] [A] [G] [E] [=] [S] [W] (") [S] [O] (;) [D] [/] [A] [EXE]

<13> [1] [2] [0] [S] [J] (IF) [B] [=] [A] [S] [O] (;) [B] [=] [0] [S] [I] (:) [C] [=] [C] [+] [1] [EXE]

<14> [1] [3] [0] [S] [H] (GOTO) [3] [0] [EXE]

This completes program input.

[Program execution procedure]

(1) After pressing [EXE] at step < 14 > above, press [MODE] [0] [S] [ø] and the display will be as follows.

NUMBER OF MOVES= ?

(2) Since we are using a 5-month moving average, press [5] [EXE] and the display will be as follows.

DATA 1 ?

(3) Input the sales quantity for the month of April which is given in the example. [5] [4] [9] [EXE]

(4) If you repeat this input up to August, the display will be as follows.

AVERAGE=451.4

(5) After checking the result, press [EXE] and the computer will ask what the next data is.

(6) Hereafter, if you input data, the average for the preceding 5-month period will be displayed. If these average values are linked, the entire trend pattern can be plotted and observed. By extending the trend line formed by linking the average values, the target product quantity can be predicted.

1.7 I want to use the PC-4 for data management or arrangement!

Suppose we have 7 products from A through G and the prices are as follows.

A = $30	D = $250	F = $80
B = $180	E = $90	G = $130
C = $170		

We can determine the weekly sales ranking for these products and arrange the figures in descending order.
In other words, this is work where data is input and, at a certain point, a computation is performed to evaluate the data and to arrange it. This operation is very similar to Multiple Item Sales Ranking which is often performed in business. Now let's process the above example using BASIC.

Sales ranking of different products

Unit price	A $ 30	B $ 180	C $ 170	D $ 250	E $ 90	F $ 80	G $ 130
Unit sales for the 1st week of September	18	6	4	1	9	20	5

It is more practical to accumulate the monthly sales for each product by lining the data up vertically. For beginners, however, we will use a simple example. With BASIC we will calculate the sales for each product from A through G and also rank them in descending order.

This is an example for those who say, "I want to use it for data management and arrangement." Managing and arranging can make data "come alive" and be more meaningful. Do not simply rely on your feelings. BASIC gives you the power to compute and analyze.

Sales ranking program

```
P0  10 VAC
    20 A=A+1
    30 INPUT "ITEM",E$
       (A)
    40 IF E9(A)="0" TH
       EN 70
    50 INPUT "UNIT PRIC
       E=",Z(A),"AMOUN
       T",Z(A+13)
    60 GOTO 20
    70 B=0
    80 FOR C=1 TO A-2
    90 IF Z(C)*Z(C+13)
       ≥Z(C+1)*Z(C+14)
        THEN 130
   100 D$=E$(C):E$(C)=
       E$(C+1):E$(C+1)
       =D$
   110 D=Z(C):Z(C)=Z(C
       +1):Z(C+1)=D
   120 D=Z(C+139):Z($C
       +13)=Z(C+14):Z(
       C+14)=D:B=1
   130 NEXT C
   140 IF B=1 THEN 70
   150 FOR C=1 TO A-1
   160 IF E$(C)="0":EN
       D
   170 PRINT E$(C),"AM
       OUNT=";Z(C+13)
   180 PRINT "$";Z(C)*
       Z(C+13)
   190 NEXT C
```

[Explanation]

Let's challenge a program which is a little longer.

It is important that you press the keys correctly while inputting the program. Relax and take your time.

If an error indication (ERR2 P0-Line Number) is displayed during program execution (RUN), perform the following correction routine operation.

[AC] [MODE] [1] [S] [LIST] line number [EXE]

The program will be displayed. Move the cursor to the desired location using the direction keys ([⇦] and [⇨]) and input the correct information, then press the [EXE] Key.

Note

◇ Program correction ◇

(1) To delete one character, perform the above operation then press the [DEL] (delete) Key.

(2) To insert one character, make an open space by pressing [S] [INS/DEL].

(3) To change information on a whole line, press line number .... [EXE]. Previous information on the line is erased and new information will be stored.

To delete a whole line, press line number [EXE].

[Program input procedure]

- The key symbol ([ ]) will be omitted except where required for special operations.

〈1〉 [MODE] 1 CLEARA [EXE] [S] DEFM [V] 25 [EXE]

The display will be as follows.

***VAR : 51

〈2〉 10 VAC [EXE]

〈3〉 20 A=A+1 [EXE]

〈4〉 30 [S] INPUT [M] [S] " [W] ITEM [S] " [W] [S] , [P] E [S] $ [R] [S] ([T] A [S]) [Y] [EXE]

〈5〉 40 [S] IF [J] E [S] $ [R] [S] ([T] A [S]) [Y] = [S] " [W] 0 [S] " [W] [S] THEN [K] 70 [EXE]

〈6〉 50 [S] INPUT [M] [S] " [W] UNIT PRICE= [S] " [W] [S] , [P] Z [S] ([T] A [S]) [Y] [S] , [P] [S] " [W] AMOUNT = [S] " [W] [S] , [P] Z [S] ([T] A+13 [S]) [Y] [EXE]

〈7〉 60 [S] GOTO [H] 20 [EXE]

〈8〉 70 B=0 [EXE]

〈9〉 80 [S] FOR [S] C=1 [S] TO [D] A−2 [EXE]

〈10〉 90 [S] IF [J] Z [S] ([T] C [S]) [Y] * Z [S] ([T] C+13 [S]) [Y] [S] < [+] Z [S] ([T] C+1 [S]) [Y] * Z [S] ([T] C+14 [S]) [Y] [S] THEN [K] 130 [EXE]

〈11〉 100 D [S] $ [R] =E [S] $ [R] [S] ([T] C [S]) [Y] [S] : [I] E [S] $ [R] [S] ([T] C [S]) [Y] =E [S] $ [R] [S] ([T] C+1 [S]) [Y] [S] : [I] E [S] $ [R] [S] ([T] C+1 [S]) [Y] = D [S] $ [R] [EXE]

〈12〉 110 D=Z [S] ([T] C [S]) [Y] [S] : [I] Z [S] ([T] C [S]) [Y] = Z [S] ([T] C+1 [S]) [Y] [S]

[I] Z [S] [T] C+1 [S] [Y] = D [EXE]

‹13› 120 D=Z [S] [T] C+13 [S] [Y] [S] [I] Z [S] [T] C+13 [S] [Y] =Z [S] [T] C+14 [S] [Y] [S] [I] Z [S] [T] C+14 [S] [Y] = D [S] [I] B=1 [EXE]

‹14› 130 [S] [NEXT G] C [EXE]

‹15› 140 [S] [IF J] B=1 [S] [THEN K] 70 [EXE]

‹16› 150 [S] [FOR S] C=1 [S] [TO D] A−1 [EXE]

‹17› 160 [S] [IF J] E [S] [$ R] [S] [T] C [S] [Y] = [S] [W] 0 [S] [W] [S] [O] END [EXE]

‹18› 170 [S] [PRINT L] E [S] [$ R] [S] [T] C [S] [Y] [S] [P] [S] [W] AMOUNT= [S] [W] [S] [O] Z [S] [T] C+13 [S] [Y] [EXE]

‹19› 180 [S] [PRINT L] [S] [W] [S] [$ R] [S] [W] [S] [O] Z [S] [T] C [S] [Y] * Z [S] [T] C+13 [S] [Y] [EXE]

‹20› 190 [S] [NEXT G] C [EXE]

[Program execution procedure]

(1) After pressing [EXE] at step < 20 > above, press [MODE] [0] [S] [0] and the display will be as follows.

ITEM ?

(2) Product A in the example is input by pressing [A] [EXE]. The display will be as follows.

UNIT PRICE= ?

(3) Press [3] [0] [EXE]. The display will be as follows.

AMOUNT= ?

(4) Press [1] [8] EXE . The display will be as follows.

ITEM ?

Input data by repeating steps (2) through (4).

(5) After inputting 5 as the amount for product G, if keys [0] [EXE] are pressed in response to ITEM?, the data will be sorted internally and the item with the highest sales value will be displayed, in this case, F.

If [EXE] is pressed again, the display will be as follows.

AMOUNT= 20

If [EXE] is pressed again, the display will be as follows

$ 1,600

Hereafter, if you keep pressing [EXE], the display will show the other data in descending order.

1.8 I want to use it as a notebook!

Refer to the part of the library section entitled "Scheduler". The PC-4 can be used as a notebook by storing a schedule, calling it out, and adding or deleting as needed.

A maximum of 22 items can be stored. If a RAM Expansion Pack (optional equipment) is added, this can be increased to 150 items.

The 26 Keys from A through Z become the drawers of the memory and you can store the desired contents in them. When more than 26 drawers are required, press

[S] [V] .

The display will be as follows.

DEFM _

If you want to make 25 more drawers, press the following Keys.

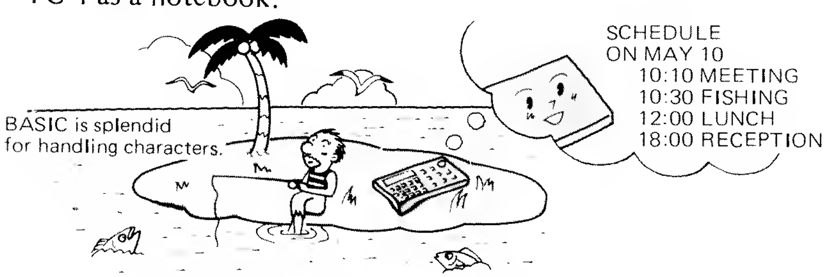
[2] [5] [EXE]

The display will be as follows.

***VAR : 51

This means you now have 51 drawers in which to store items.

The [V] Key plays an active role when using the PC-4 as a notebook.



For those who want to use the PC-4 as a notebook, try the following program. This is a program for inputting items and prices one after another. Then later, the amount and pricing data for a certain item can be called out. This is also a convenient program to register those in attendance at a certain event, such as charity concerts, charity bazaars, etc.

Program for items and prices

```
P0
  10 INPUT "ITEM",C$
  20 FOR B=1 TO 23
  30 IF C$=D$(B) THE
     N 60
  40 NEXT B
  50 PRINT "NO NAME"
     :GOTO 10
  60 PRINT "PRICE=";
     Z(B)
  70 INPUT Z(B)
  80 GOTO 10
P1
  10 C=0
  20 FOR B=1 TO 23
  30 C=C+Z(B)
  40 NEXT B
  50 PRINT "TOTAL=";
     C
P9
  10 VAC
  20 FOR B=1 TO 23
  30 INPUT "ITEM",D$
     (B)
  40 INPUT "PRICE",Z
     (B)
  50 NEXT B
```

[Explanation]

The PC-4 permits storage of 10 different programs in program areas P0 through P9. This allows one program to be divided and stored in three different areas for individual purposes. For example, the data may be input into P9, called out using P∅, and totalled using P1.

[Program input procedure]

The key symbol (□) will be omitted except where required for special operations.

< Input to P9 >

<1> [MODE] 1 CLEAR A [EXE] [S] [P9 9] [S] [DEFM V] 23 [EXE]

The display at this time will be as follows.

***VAR : 49

<2> 10 VAC [EXE]

<3> 20 [S] [FOR S] B=1 [S]

[D] 23 [EXE]

〈4〉 30 [S] [M] [S] [W] ITEM [S] [W] [S] [P] D
[S] [R] [S] [T] B [S] [Y] [EXE]

〈5〉 40 [S] [M] [S] [W] PRICE [S] [W] [S] [P]
Z [S] [T] B [S] [Y] [EXE]

〈6〉 50 [S] [G] B [EXE]

< Input to P0 >

〈1〉 [S] [0] 10 [S] [M] [S] [W] ITEM [S]
[W] [S] [P] C [S] [R] [EXE]

〈2〉 20 [S] [S] B=1 [S] [D] 23 [EXE]

〈3〉 30 [S] [J] C [S] [R] =D [S] [R] [S] [T] B [S]
[Y] [S] [K] 60 [EXE]

〈4〉 40 [S] [G] B [EXE]

〈5〉 50 [S] [L] [S] [W] NO NAME [S] [W]
[S] [I] [S] [H] 10 [EXE]

〈6〉 60 [S] [L] [S] [W] PRICE = [S] [W] [S]
[O] Z [S] [T] B [S] [Y] [EXE]

〈7〉 70 [S] [M] Z [S] [T] B [S] [Y] [EXE]

〈8〉 80 [S] [H] 10 [EXE]

< Input to P1 >

〈1〉 [S] [1] 10 C=0 [EXE]

〈2〉 20 [S] [S] B=1 [S] [D] 23 [EXE]

〈3〉 30 C=C+Z [S] [T] B [S] [Y] [EXE]

〈4〉 40 [S] [G] B [EXE]

〈5〉 50 [S] [L] [S] [W] TOTAL = [S] [W] [S]
[O] C [EXE]

[Program execution procedure]

(1) Press [MODE] [0] [S] [9] and the display will be as follows.

ITEM ?

(2) For example, input HOLMES and press [EXE] and the display will be as follows.

PRICE ?

(3) For example, input 20 and press [EXE] and the display will be as follows.

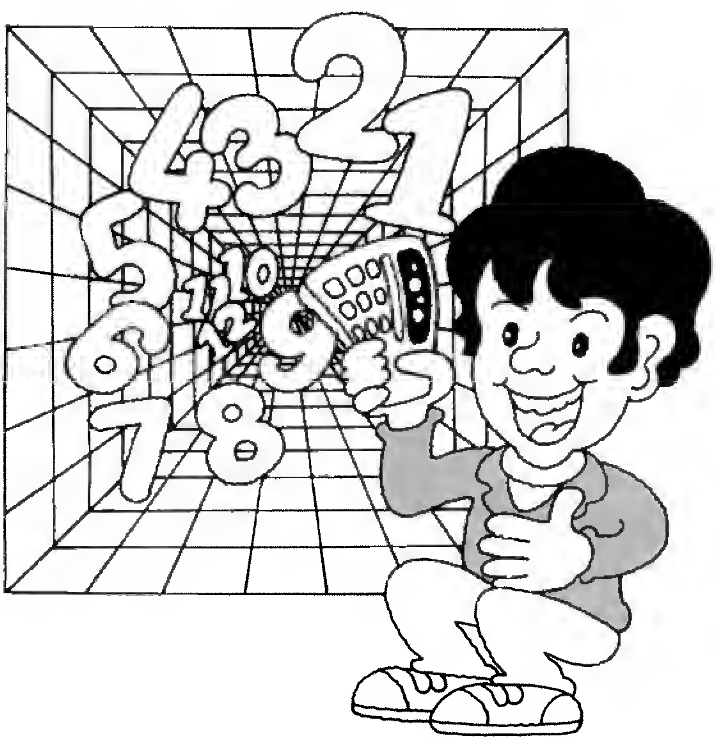
ITEM ?

Input various names and prices one after another. A total of up to 23 names and prices can be input.

(4) For example, if you want to know the data for HOLMES, press [S] [0] [H] [O] [L] [M] [E] [S] [EXE] and the price will be displayed. Also, the total can be obtained by pressing [S] [1].

Chapter 2

What Is a Program?



2.1 The Radio Shack PC-4, a pocketable computer and calculator in a single unit

The words "program" or "programming" are usually associated with computers. But what exactly is a program? Simply, a program is a set of instructions telling a computer what to do. For those studying programming for the first time, we will discuss how to think in terms of the program language BASIC. This does not mean that we will be speaking about something which is very difficult to understand, so just relax and read on.

First, let's take a look at the Radio Shack PC-4 and how it operates.
The PC-4 is a real computer but may also be used as a calculator.
First, turn the Power Switch on.

```
   RUN
      DEG
READY  P0
```

The display on the PC-4 will appear as shown above. Press the following keys in sequence.

[3] [+] [5] [EXE]

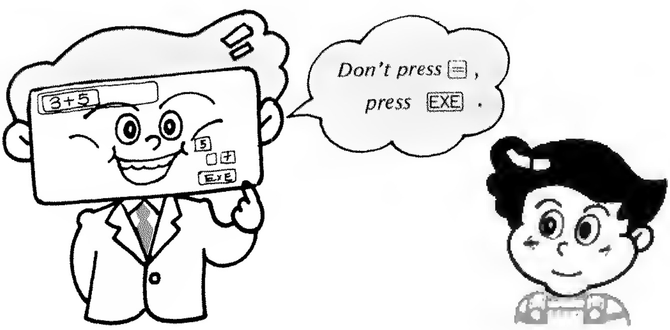
It will take a moment to locate the positions of the keys at first but,after using the PC-4 for a while, you will be able to find them quickly.
The EXE Key is located on the lower right side of the PC-4. EXE is short for EXECUTE and it acts like an equals sign (=) Key on an ordinary calculator.

You should get a result of 8 after you press this key.
Those who are used to performing addition on a calculator may consider this procedure a little strange at first. That's because when using a standard calculator the operation is completed by pressing the following keys.

[3] [+] [5] [=]

If you try to perform the operation in the PC-4 by using this sequence, you will get a blinking "–" (This is called the "cursor".) instead of an answer.
The [=] Key on the PC-4 is only used during programming and has a different meaning. With the PC-4, the [EXE] (Execute) Key is used instead of the [=] Key for calculations.

The [EXE] Key is also pressed to execute a command or input data.



By the way, a standard calculator has an [M+] Key (Memory Plus Key) and an [MR] Key (Memory Recall Key) to store numbers in the memory and to call out the memory contents. If you take a look at the PC-4 Keyboard, you will not find these Keys. You might think that the PC-4 has no memory. However the PC-4 is a powerful computer, with superior memory functions, when compared to a calculator.

Calculator memory and PC-4 memory.

For example, press the following Keys.

A EXE

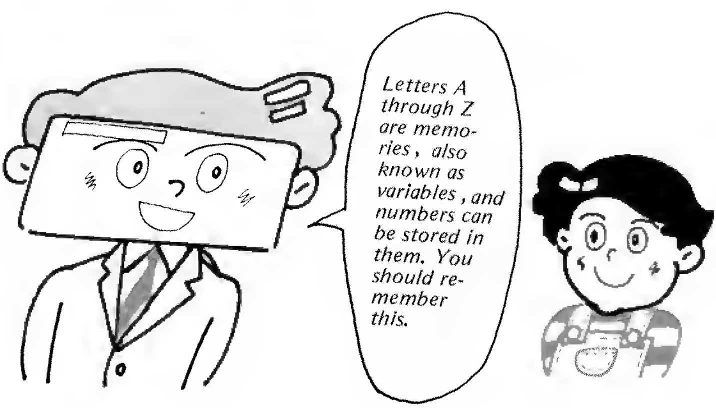
A number will appear on the display.
That number is the contents stored in memory "A". The above key operation simply calls out the contents of memory "A". Now press the following keys.

Z EXE

Is there any display?
This operation calls out the contents of memory "Z". In this manner, you can see that there are 26 memories from A to Z in the PC-4.

In a program, memories A to Z are called "variables."

Numbers with a maximum of 12 digits can be stored in each memory and a maximum of 10 digits can be displayed at one time.
Memories A to Z are known as "variables" when they are used in programming.



These memory contents can be used for performing addition, subtraction, multiplication and division when they contain known numbers.
For example, press the following keys.

A = 1 0 EXE and Z = 2 EXE

These operations store 10 in memory "A" and 2 in memory "Z". Note the previously stored numbers have been replaced by 10 and 2.
Now press the following keys.

A + Z EXE

The result should be 12.
Next, press the following keys.

A − Z EXE

What is the result now? It should be 8.
Now let's perform multiplication and division.
If you look at the PC-4, you will see that there is no × or ÷ Key.
Since the symbol for multiplication looks like the letter "X", an asterisk (*) symbol is used instead. Also, a / (slash) symbol is used for division instead of the conventional "÷" symbol.
Now let's try multiplication and division using the contents of memory "A" and memory "Z".
As you may have guessed, the key operation is as follows.

You are slowly getting used to operating the Keys.

A * Z EXE
A / Z EXE

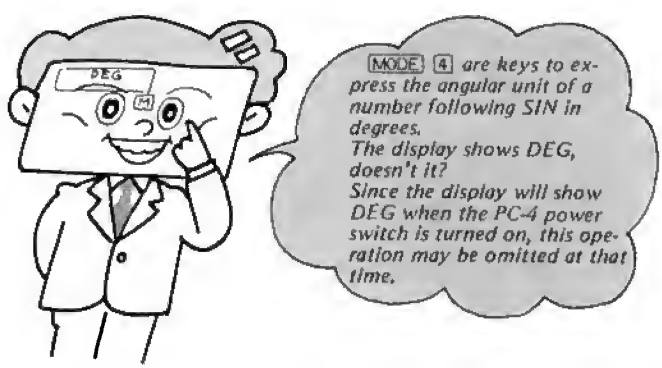
Were the results 20 and 5 respectively?
Incidentally, if you make a mistake and touch the wrong key, you don't have to clear everything. Press the direction keys (⇦ and ⇨) and enter the correct number.

2.2 There are no scientific function keys on the PC-4 but don't worry.

The PC-4 has 26 memories (from A to Z) and we have learned that we can use these when performing addition, subtraction, multiplication and division.
Now we will learn how to perform other functions such as sine, cosine and tangent.
Some people might say, "I can do those kinds of calculations quickly using a scientific calculator." Scientific calculators have keys such as [sin], [cos] and [tan]. The PC-4 does not have these keys.
Let's try these function calculations using the PC-4.
For example, obtain sin 30°.
Key operation is as follows.

[MODE] [4] …… [illegible]
[S] [I] [N] [3] [0] [EXE]

The result is displayed as 0.5.



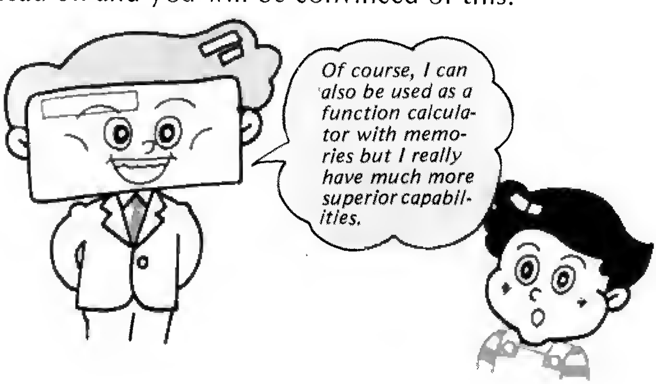
You must pay attention to one point. With a function calculator, input is made as follows.

[3] [0] [SIN]

But with the PC-4, the function is entered first.
This is very important when programming so please remember this point.
Now, let's use the knowledge gained so far and use the memory contents instead of the number 30.
Memory "A" now contains the number 10, doesn't it?
Press the following Keys.

[S] [I] [N] [A] [EXE]

The result should be 0.1736481777.
As you know by now, the above is the method for obtaining sin 10°.
From this you can see that the PC-4 not only performs all of the functions of a scientific calculator but also has abundant memory to facilitate those functions.
These functions are just a small portion of the many capabilities of the PC-4.
Read on and you will be convinced of this.



2.3 Obtaining the total of the numbers 1 through 10

We have already explained that the ability to perform scientific function calculations with abundant memory functions is only one of the many capabilities of the PC-4. Well, what other capabilities does the PC-4 have? It is capable of computing functions using a variety of programs.

If you do not know much about programming, you might think this function is of no use to you.

But by using a system of simple "manual programming", you will slowly be led into the knowledge of programming computers. Now let's obtain the cumulative totals of the numbers 1 through 10. Actually, if you are good at mathematics, you can get the answer pretty quickly on your own without using a computer but bear with us for a moment.

- **The first method. The calculator approach.**

[AC] [1] [+] [2] [EXE]	3
[+] [3] [EXE]	6
[+] [4] [EXE]	10
(Continue in the same manner for 5 through 8)	
[+] [9] [EXE]	45
[+] [1] [0] [EXE]	55

The above key operation is one method of obtaining the incremental totals of the numbers 1 through 10.

This operation is almost the same as that used for a

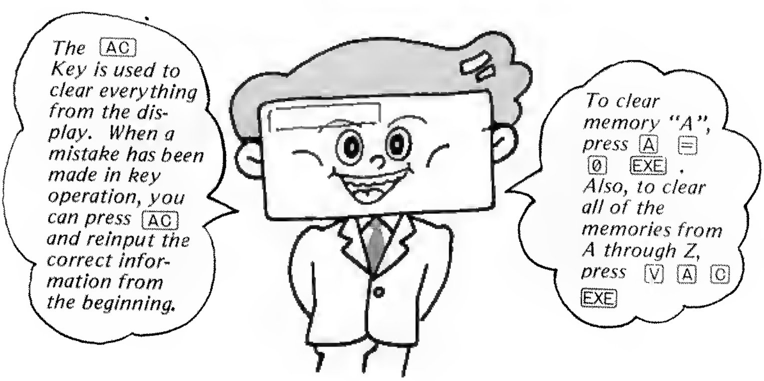
standard calculator.

- **The second method. A bit closer to actual programming.**

Remember that the PC-4 has memories from "A" to "Z".
Using one of these memories, we can obtain the total in a different manner.
Here we come a bit closer to thinking in terms of a program.
Turn the power switch on.
Since we will be using memory "A", clear it by pressing the following Keys.

[A] [=] [Ø] [EXE]

This operation means "write Ø to memory A" and the [EXE] Key means "execute this operation."



Now let's check to see if the above operation stored a Ø in memory "A".

Press the following Keys.

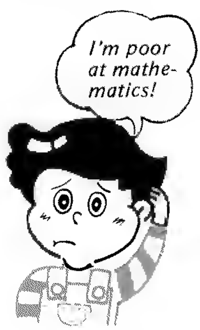
[A] [EXE]

Is a Ø displayed?
Next, we will add the numbers 1 through 10 in sequence in memory "A".
First, to add in the number 1, press the following Keys.

[A] [=] [A] [+] [1] [EXE]

The [=] symbol is different from the equal sign used in mathematics.

The equal symbol does not mean that the two sides of the equation are equal. It merely means "put the result of the operation on the right side of the [=] symbol into memory A". The [=] symbol used here has a meaning which is different from the [=] sign used in mathematics. Since it is easy to confuse the meaning of the [=] symbol, please be careful.



Now let's check the contents of memory "A" to see what it has become as a result of performing the above operation.
Press the following Keys.

[A] [EXE]

Is the result 1?
In the same manner, repeat the operation the required number of times.

[A] [=] [A] [+] [2] [EXE]
[A] [=] [A] [+] [3] [EXE]

(Continue in the same manner for 4 through 9.)

[A] [=] [A] [+] [1] [Ø] [EXE]

Finally, call out the contents of memory "A" ···· [A] [EXE] ···························· 55 Total of the numbers 1 through 10

It was a bit lengthy, but did you get the correct answer?
You must be careful not to omit anything and not to add the same number twice.
Now let's try a third method.

● **The thrid method. Even closer to actual programming.**

Since the PC-4 has 26 memories (from "A" to "Z"), let's prepare another memory and see what numbers were just added.

Since the numbers which were added to one another increase by 1 in sequence, we will store this in memory "I".

Clear memory "I" in the same manner as "A" was cleared in the previous example.

Press the following keys.

[A] [=] [0] [EXE]
[I] [=] [0] [EXE]

Then add 1 to the contents of memory "I".

[I] [=] [I] [+] [1] [EXE]

Check to see if the operation was successful.
Press the following keys.

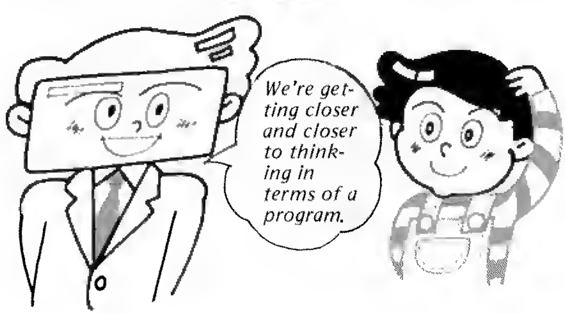
[I] [EXE]

Is the result 1?

Next, add the contents of memory "I" to memory "A".

Press the following keys.

[A] [=] [A] [+] [I] [EXE]



The entire operation is as follows.

[A] [=] [0] [EXE]
[I] [=] [0] [EXE] — This clears memory "A" and memory "I".

Adds 1 to the "I" on the left of the = symbol — [I] [=] [I] [+] [1] [EXE]
Adds I to the "A" on the left of the = symbol — [A] [=] [A] [+] [I] [EXE] — 1st time

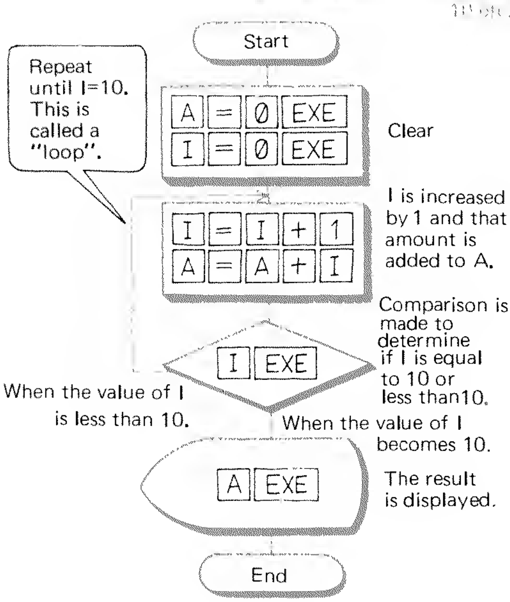
Adds 1 to the "I" on the left of the = symbol — [I] [=] [I] [+] [1] [EXE]
Adds I to the "A" on the left of the = symbol — [A] [=] [A] [+] [I] [EXE] — 2nd time

Continue until the I and A cycle has been performed 10 times.

[I] [=] [I] [+] [1] [EXE]
[A] [=] [A] [+] [I] [EXE] — 10th times

[A] [EXE]

The sequence makes the "A" on the left of the = symbol sequentially equal to 1, 3, 6, 10 etc. up to 55.



Using this operation method, if the [I] and [EXE] Keys are pressed along the way, you can check to see what number is being added at that point.

If you take a look at the figure on the left, the Key operation which was listed above will be easier to understand.

2.4 Simultaneous calculation of sum and square sum

In the previous examples, the total of the numbers from 1 through 10 was written in memory "A". Now let's extend the operation further and also write the square sum from 1 to 10 in memory "B". In this manner, two jobs can be combined into a single sequence.
This means that both of the following operations will be executed simultaneously.

$A \leftarrow 1+2+3 \;.....\; +10$

$B \leftarrow 1^2+2^2+3^2 \;.....\; +10^2$

- **The fourth method. We are now beginning to really resemble programming.**

If this operation is performed using a calculator with an ordinary memory, A will be obtained first and B will be obtained last.
However, if you use a method which resembles programming as shown below, A and B can be obtained simultaneously.

Input

[A] [=] [0] [EXE]

[B] [=] [0] [EXE]

[I] [=] [0] [EXE]

[I] [=] [I] [+] [1] [EXE]

[A] [=] [A] [+] [I] [EXE]

[B] [=] [B] [+] [I] [*] [I] [EXE]

I increases by 1 — I = I + 1 EXE

The I which has been increased by 1 is added to the former contents of A, and A becomes a new value. — A = A + I EXE

The I which has been increased by 1 is squared and this is added to the former contents of B, and B becomes a new value. — B = B + I * I EXE

A EXE

B EXE

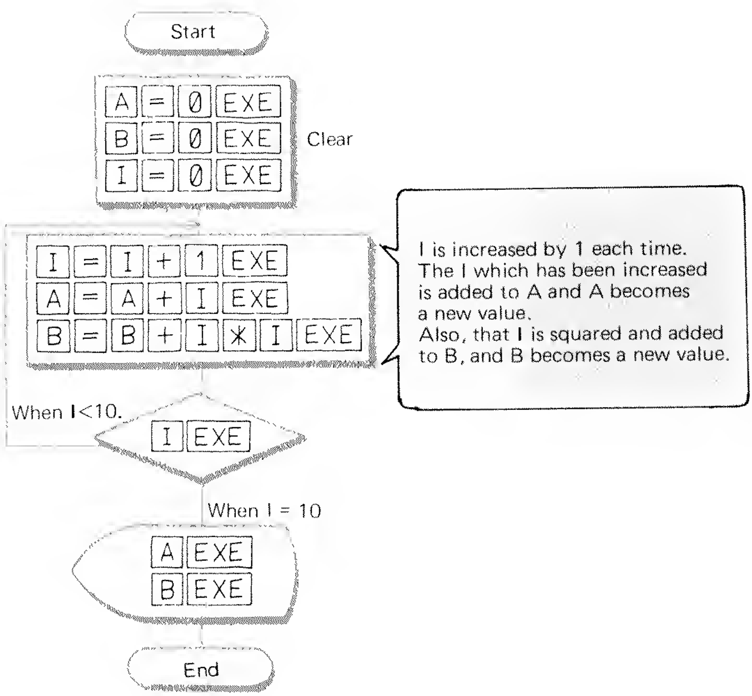
Displays the final value of A and B.

To find out the current step you are on during the above operation, simply press

I EXE

As with the third method, this operation will be easier to understand if you refer to the figure below.

When computers were less functional and more expensive, procedures such as that shown below were time consuming and cumbersome.



Also, the keying operation requires much patience. The "programming" approach enables us to put the entire procedure into the memory and simply plug in the desired numbers.

● **The fifth method. Extremely close to programming.**

Now, let's try just a little harder and expand the way of thinking even more.

In the fourth method, the sum of the numbers 1 through 10 was obtained as well as the sum of the squares. Now we would like to obtain the sum of optional 10 numbers as well as the square sum.

At this time, there is a new requirement to store the value of the optional numbers in a memory. Let's use memory "X" for this purpose. Memories "A", "B" and "I" are used the same as before. Memory "I" is used to count the number of times the optional number values (called data) are input.

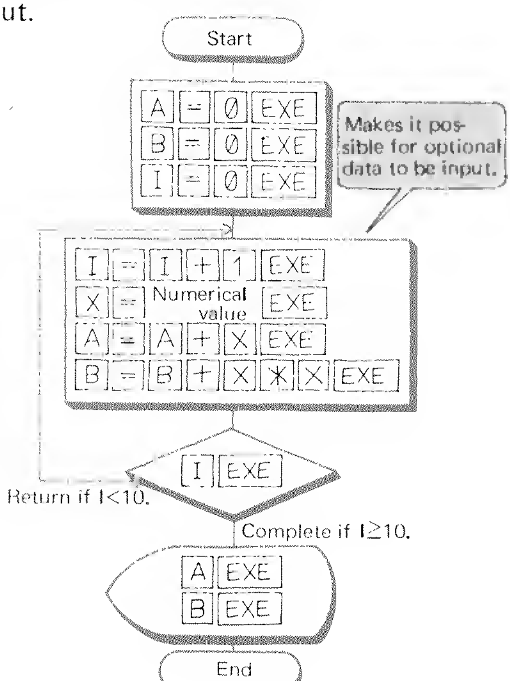
If the input value (data) is labeled as Xi (i = 1, 2 ..... 10), we get the following.

$A \leftarrow X_1 + X_2 + \dots + X_{10}$

$B \leftarrow X_1^2 + X_2^2 + \dots + X_{10}^2$

As with the third and fourth methods, the entire procedure is as shown on the right.

Try to operate the keys in accordance with the figure and input some actual data.



By the way, this example is a very useful one. Suppose the data of X_1 through X_{10} are test results.
The average score can be obtained in the following manner.

[A] [/] [I] [EXE]

In Chapter 4, Section 5, entitled "Challenging the troublesome deviation values", will explain some useful applications for the square sum.
Following is some sample data. Use it to obtain the average score.

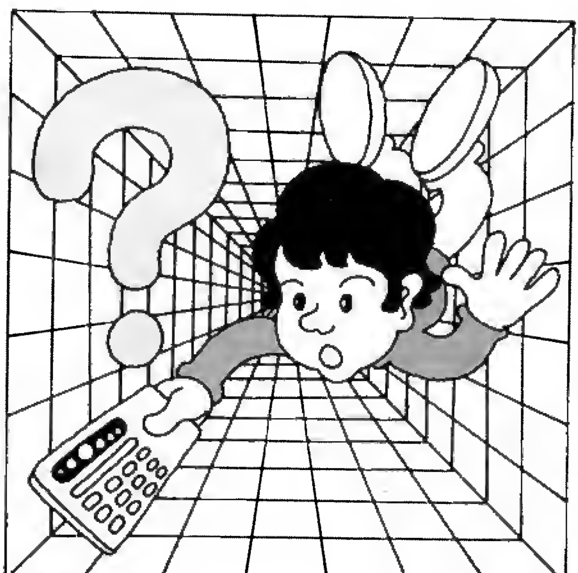
30, 50, 80, 40, 20, 70, 60, 90, 50, 80

If the value of A becomes 570 and the value of B becomes 37300, the answer is correct and the average score will be 57.
How did you make out? Did you make the inputs correctly?
Was it difficult?

The operations that have been performed until now have brought us to a point where our thinking approximates actual programming. Nevertheless, it is still "manual programming", similar to using a calculator.
At this point, you are probably tired of using so much key input so we will stop "manual programming". In the next chapter, we will perform programming using the same values which were used in this chapter. You will be surprised at how clear-cut the methods are.

Chapter 3

This Is a Program!



3.1 Programming at last !

Now let's get into some actual programming. The language used for programming the PC-4 is BASIC, one of the most popular computer programming languages.
Even such a compact pocketable computer as the PC-4 can still make splendid use of BASIC. In addition, the PC-4 can use unabbreviated words such as PRINT and INPUT, and has the same kind of performance as a full-sized personal computer.
This is why the PC-4 is very useful for beginners who are just learning BASIC.
The BASIC language itself will be explained later in detail in Chapter 4. In Chapter 3, we will learn how to make actual programs to obtain the same results noted in methods three through five in Chapter 2.
Relax and enjoy Chapter 3.

● **Before starting, please note a few points.**
Since the terms and symbols will be slightly different from those previously used, you should note the correct usage. First of all, in the "manual programming mode" used in the preceding Chapter, the "A" through "Z" Keys were explained as if they were ordinary calculator memories. They were referred to as memory "A", memory "B", etc.
In a program, however, they will be called "variables". For example, they are referenced as "variable A",

"variable B", etc. So even if referred to as "A" and "B", please think of them in terms of being variables with numerical values.

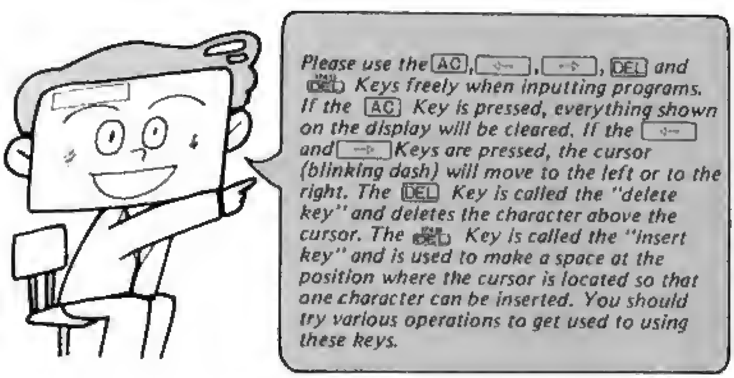
Also, in executing an operation such as inputting 10 into variable "A", we press the following keys.

A variable is a symbol, such as "A", whose value may be any number such as 10, 20, etc.

[A] [=] [1] [0] [EXE]

In this case, [EXE] means to execute the contents written to the left of [EXE] . It is a fundamental requirement that this key be pressed at the end of each line.

However, the changing of mode or program area can be done without pressing the [EXE] Key. Key expressions such as [A] and [=] will simply be written as A and = when the meaning is clear.



3.2 Program to obtain the total of the numbers 1 through 10

Let's actually program the operation to perform the computation which was done in method three on page 46 using "manual programming".
Turn the PC-4 Power Switch on.

```
   RUN
      DEG
READY  P0
```

RUN mode

Does your PC-4 show a display like the one above. The important thing here is that RUN is displayed. This means that the PC-4 is in the RUN mode. Actually, all of the previous examples were executed in the RUN mode.
(While we are on the subject, DEG was the degree mode of the angular unit, wasn't it?)
But, since we now want to write a program, we must change the mode.
Press the following Keys.

[MODE] [1]

```
      Blinks
               DEG
         WRT
P   0 1 2 3 4 5 6 7 8 9
```

If the 0 does not blink, press [C][L] [E][A][R][A][EXE]

The display will be as shown above. The RUN mode has been cleared and WRT appears. This is an abbreviation for the word "WRITE" and it indicates that the PC-4 is in a WRITE mode which permits a program to be written.
Referring to the figure on Page 46, write in the key operation just as it is shown. Write in the first two lines.

A=0 [EXE]

I =Ø EXE

The cursor (blinking bar) will appear on the left side of the display.
Now let's check to see if the two lines which have been input are actually stored in the PC-4.
To do this, press the following Keys.

L I S T EXE

This command may be input using the individual alphabet Keys to spell out LIST or by pressing S (the SHIFT Key) and N.
After this, the display should return to the original condition.
Is the Ø blinking?
Too bad! This means that nothing was written into the program. Read on and you will find out why.

LIST is a command to display a program which has been written and stored in the PC-4.
The S Key is known as the SHIFT Key. Most of the keys have red lettering above them. To access the words in red lettering, press S. "S" will appear on the display. Then press the desired key. "S" will disappear from the display, and the appropriate symbol or character will be displayed.

- **Line numbers are required in a program**

When actually writing a program, each line of the program must have a name, a way to address each specific part. Normally, we think of a name as being something like "John" or "Mary".
But in BASIC, we use numbers which are called "line numbers".
Any numbers from 1 to 9999 may be used but it is customary to use 10, 20, 30, etc. to allow for additions or insertions to the program.
Now let's make some inputs using line numbers.

10 A=Ø EXE
20 I=Ø EXE

At this time, line 20 should remain on the display.
Now let's check the input using the LIST function.
Press the following keys.

S LIST N EXE

Did the display show the following?

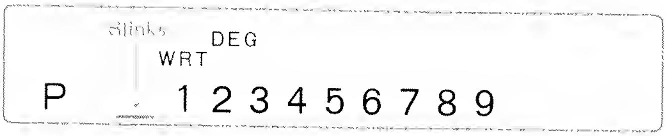
10 A=Ø

Since a program runs in sequence from low line numbers to high line numbers, start with low numbers and go toward higher numbers.

Next, if the [EXE] Key is pressed, the following should be displayed.

20 I=0

If the [EXE] Key is pressed again, the display will be as follows.



The ∅ will be missing indicating that a program is entered in this memory space. The blinking cursor will be in the ∅ position. However, the WRT symbol will remain.

In other words, the PC-4 will remain in the WRT (WRITE) mode.

Next, input the following.

30 I=I+1 [EXE]

40 A=A+I [EXE]

Please take another look at the figure on page 46. The ◇ symbol in the figure shows the location where a decision is made.

Make inputs as follows.

IF ~ THEN

50 IF I<10 THEN 30 [EXE]

This means, "If I is less than 1∅, then jump to line 3∅." This is a BASIC language command statement.

Next, to see the result of A, make the following input.

PRINT

60 PRINT A [EXE]

PRINT means to show the result on the display.

As the final step in the program, be sure to input the following.

END

70 END [EXE]

This END means that the program is finished. Even if END is not input, the program will still work. However, END is the standard way to terminate a program.

Let's list the program

Let's list the program to see if it was written in correctly. Press L I S T EXE or S N EXE . The following display should show line by line.

List I

```
10  A=0
20  I=0
30  I=I+1
40  A=A+I
50  IF I<10 THEN 30
60  PRINT  A
70  END
```

When listing the program, if a line is long, such as line 50 in this example, press the Key. Each time it is pressed it moves the characters one position to the left on the display so that you can see the entire line.

If the list shows on the display just as in the above example, the program is correct. Also, if a program is listed in the WRT mode, only one line is displayed at a time. You must press the EXE Key to display the next line.

If a mistake has been made on a line, press the direction keys (⇦ or ⇨) to move the cursor to the left or right until it is beneath the character to be corrected, then input the correct character.

At this point, program input is complete. Compare the amount of key operation required to perform the operation on page 46 with the amount required here. Just this small amount of work here accomplishes exactly the same results.

Let's execute the program we have just written.

Program execution is performed in the RUN mode.
Press the following keys.

MODE 0

MODE 0 designates the RUN mode.
After doing this, the display will be the same as it was when the power switch was turned on. The display will be as shown below.

```
RUN
  DEG
READY P0
```

The P0 program can also be executed by pressing S 0.

The RUN mode and the RUN used to execute a program are different so please be careful.

This display means that program P0 can be executed. At this time, input one of the following to execute program P0.

R U N EXE or S RUN 0 EXE

After about one second, the result of 55 will be displayed and STOP will be displayed on the upper right of the display. This means that the program execution is complete. The troublesome operation which before required manual key inputs to be made 10 times is executed easily with this simple program.
And yet, "repetition of the same key operation has been replaced by only one input". This is very different from an ordinary calculator, isn't it?
This is the power of the PC-4.

3.3 A slight change in the program

Let's take a closer look at the power of the PC-4. In the previous section, the total of the numbers 1 through 10 was obtained. Now the total of the numbers from 1 through 100 will be obtained.
This program will be a slightly modified version of the previous program.
Put the PC-4 back in the WRT mode.
To do this, input [MODE] [1].
The display will be as follows.

```
              DEG
          WRT
  P    _   1 2 3 4 5 6 7 8 9
```

The blinking cursor will appear where the Ø should be located. This means that the PØ program area already has a program written in. Also, the blinking cursor means that it is possible to write data to program area PØ.
List the program to check it.
At this point, we want to change the program from 1 through 10 to 1 through 100. All we need to do is to change 10 to 100 in the decision portion shown by the ◇ symbol on page 46.
Display line 5Ø by inputting the following.

[L] [I] [S] [T] [5] [0] [EXE] or [S] [LIST N] [5] [0] [EXE]

Line 5Ø will be displayed.
Then, using the direction key ([⇦]) and the [INS DEL] Key, change the required portion.
For practice, change the line to read as follows.

```
50 IF I<100 THEN 30
```

Were you successful?
After correction, return to the RUN mode in the same manner as before and confirm that READY PØ is displayed.
Then execute the program by inputting the following.

R U N EXE or S RUN B EXE

After a few seconds, a result of 5050 will be displayed.
This kind of simplified operation is the big difference between the PC-4 and ordinary calculators.

Note

◇ Step number ◇
If the WRT mode is designated by pressing MODE 1, the display will appear as follows. (The display indicates a case where no programs are written in the program areas.)

```
WRT DEG
P Ø 1 2 3 4 5 6 7 8 9
```

If all the programs have been cleared, the "544" shown on the upper right of the display indicates the remaining number of program steps.
In this case, it means that 544 steps remain.
Since nothing is written, it indicates that a maximum of 544 steps can be written into program areas PØ through P9.
The step numbers are counted as follows.

(1) Line numbers (1 to 9999) ..... 2 steps
(2) Command names and function names (FOR, TO, SIN, PRINT, etc.) ..... 1 step
(3) Line spacing (required at the end of each line) ..... 1 step
(4) Other characters ..... 1 step per character
(5) All spaces (SPC) other than those in character strings enclosed by quotation marks (" ") are ignored.

As a program is written, you will see the step numbers on the display decrease.

3.4 Program to obtain the square sum simultaneously

Next, let's make a program to accomplish the operation performed in the fourth method on page 47. This is a calculation to obtain the total of the numbers 1 through 10 and their square sum simultaneously. The program approach is the same as that used in the previous example.
The program is shown below.

List 2

```
                           10  A=0
                           20  B=0  ..................... Clears the square
                                                          sum variable
                           30  I=0
                          *40  I=I+1
Return to here if          50  A=A+I
I is less than 10          60  B=B+I*I  ................. Square sum calcu-
                                                          lation
                           70  IF I<10 THEN 40
                           80  PRINT A
                           90  PRINT B  ................. Square sum result
                          100  END
```

The only difference between this program and the example in the previous section is that line numbers 2∅, 6∅ and 9∅ have been added.
Now, for practice, let's write the program into the PC-4.
Place the PC-4 in the WRT mode in order to write in the program. The cursor in the ∅ position should blink as before.

If this (List 2) program is written in at this time, the previously written (List 1) program will be erased.
By the way, the PC-4 has 10 locations (from P∅ through P9) where programs can be written. This means that up to 10 programs can be stored simultaneously.

WRT DEG (A total of 10 programs can be written in.)

P _ 1 2 3 4 5 6 7 8 9

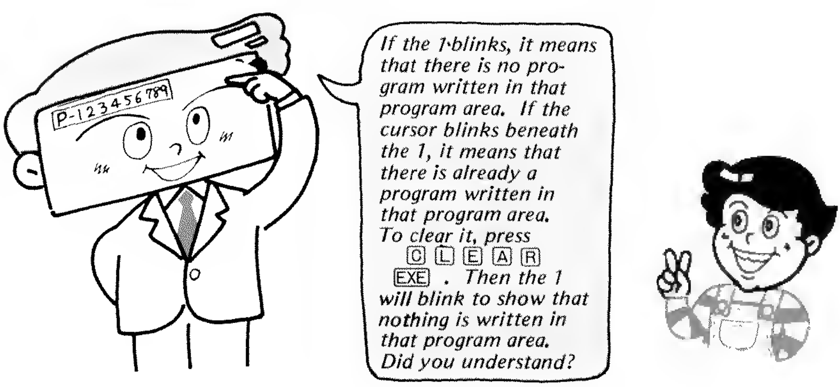
List 1 is stored here. Store List 2 here.

Now we will write List 2 into the P1 area.
The procedure is as follows.

[MODE] [1] .... Selects the WRT mode. (Cursor will blink at the P∅ position.)

[S] [1] (P1) .... Selects the P1 area (The 1 at the P1 position will blink.)

If the 1 blinks, it means a program can be written into program area P1.
If the cursor blinks instead of the 1, it means a program is already stored in P1. To erase that program, input [C] [L] [E] [A] [R] [EXE] .
This will clear the program currently stored in the P1 program area and the 1 will blink. At this time, if you attempt to list the program nothing will be displayed.



● Write-in and execution of List 2

Write the program shown as List 2 on page 61 into the PC-4.

After writing, check the LIST contents. If any errors are found, make corrections using the direction (⇦ , ⇨), delete (DEL) and insert (S INS) Keys.

Now return to the RUN mode by inputting MODE 0.

The display should be as follows.

```
READY P1
```

In computer terms, program checking and correction is known as "program debugging".

If READY P0 is displayed, input S P1.

The program will begin execution immediately.

Also, to execute when the display shows READY P1, input R U N EXE . Then 55 will be displayed.

This corresponds to program line number 80.

If the EXE Key is pressed again, 385 will be displayed.

This is the result of the execution of line number 90.

If the EXE Key is pressed again, the program will go to the END at line number 100 and the STOP display will go off.

Note

If you operate the PC-4 while reading this manual, the display may sometimes go off. This is to save battery power. The PC-4 will shut off automatically after 7 minutes have passed without any input. To turn it back on, press AC/ON.

3.5 Program for obtaining the sum of optional numbers

You have learned very important concepts by now and we hope that things are becoming more and more interesting. But don't get impatient. Take a moment now to relax.

Well now that you have had a rest, let's make a program using the figure on page 49.
This is a calculation for obtaining the sum and the square sum by inputting 10 numerical values into the variable X.
By the way, this program has one difference when compared to the previous programs.
The difference is that the numerical values must be entered whenever the display shows a question mark ("?")

INPUT

An INPUT X command is used to input numerical values into the program.
The program is as follows.

List 3

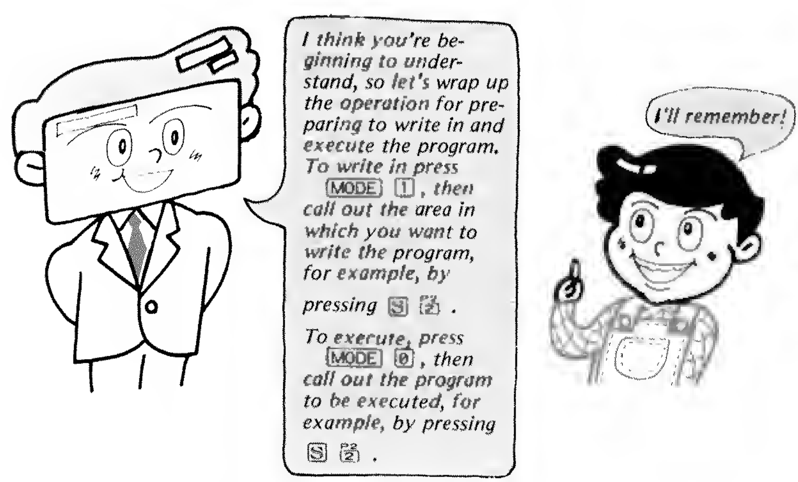
```
10 A=0
20 B=0
30 I=0
40 I=I+1
50 INPUT X
60 A=A+X
70 B=B+X*X
80 IF I<10 THEN 40
90 PRINT A
```

```
100  PRINT B
110  END
```

This program will become the P2 program. As you probably know by now, input [MODE] [1] then [S] [P2 2] The display will be as follows.

```
           DEG
       WRT
  P   _ _ 2 3 4 5 6 7 8 9
```

The 2 will blink.
If the cursor blinks in the 2 position, it means a program is already stored so erase it by using CLEAR [EXE].
After inputting the program, check for errors by listing it and return to the RUN mode by pressing [MODE] [Ø] .
The display should show READY P2. Then input RUN [EXE] and the program can be executed.
This procedure has already been performed many times before so you have probably become skilled by now.
If READY PØ or READY P1 appears, call out P2 by pressing [S] [P2 2] .
Program execution will begin immediately.



When the program executes line number 4∅, a "?" will be displayed. This means "input the data for variable X". For example, if 3∅ is input, the display will be as follows.

30

If the EXE Key is pressed, the "?" will be displayed again. Continue to make inputs in the following sequence.

5∅, 8∅, 4∅, 2∅, 7∅, 6∅, 9∅, 5∅, 8∅

After completing input of the 10 data elements, the sum A, a result of 570, will be displayed.

Next, if the EXE Key is now pressed, the square sum B, a result of 37,300, will be displayed.

Note

There are two ways to stop program execution. One method uses the STOP Key and the other method used the AC Key. If the STOP Key is used, STOP will appear on the upper right of the display. If the STOP Key is used again, the program area and line number where the program execution was stopped will be displayed. For example, P∅ – 7∅. Then if the EXE Key is pressed, program execution will restart from the stopped location.

On the other hand, if the AC Key is used, program execution will stop. When you desire to stop the program execution completely use this method.

*There may be times when the STOP Key and AC Key do not produce the desired results. In that case, use the power switch to stop the program.

3.6 A slight change in the progam

By making modifications in a program, many different programs can be created.

For example, the List 3 program in the previous section had 10 different data elements. It can be changed as follows.

- **When the number of data is known but yet it may vary.**

Change the program so that the number of data elements can be input into variable N and also change the decision portion on line 8∅ from 1∅ to N. The program is as follows.

List 4

```
10  A=0
20  B=0
30  I=0
40  PRINT"N=";
50  INPUT N          ← Input number of data
60  INPUT X
70  A=A+X
80  B=B+X*X
90  I=I+1            ← Input data count
100 IF I<N THEN 60
110 PRINT A
120 PRINT B
130 END
```

Returns to here if I is less than N.

The approach is changed from 10 entries to N entries for the variable.

If we compare List 3 and List 4, we see that the position where the input data is counted is different.
This position can be anywhere between line number 6Ø and line number 9Ø. Line 4Ø and line 5Ø can be written on the same line using INPUT statement:

```
40 INPUT "N=", N
```

● **When the number of data elements is not known.**
It is sometimes troublesome to count the number of input data ahead of time. Then it becomes difficult to determine how to get the result.
For example, if it is known that the value of the data will not be a negative (minus) value, there is a method for inputting a negative value on purpose to terminate the process. This method is often performed. The program is as follows.

Write this program and execute it.

List 5

```
10  A=0
20  B=0
30  I=0
40  INPUT X
50  IF X<0 THEN 100
60  I=I+1
70  A=A+X
80  B=B+X*X
90  GOTO 40
100 PRINT A,B
110 END
```

Return

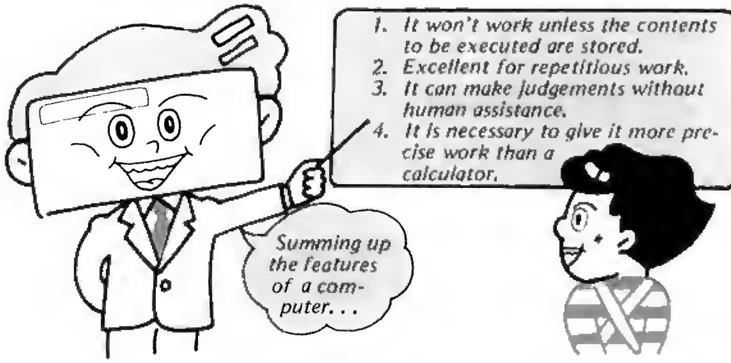
Jump

PRINT A and PRINT B can be written on the same line by separating A and B with a ",".

The GOTO on line number 9Ø means "transfer execution to line number 4Ø".

3.7 Conclusion

Chapters 2 and 3 have explained how to perform calculations on the PC-4 by using a calculator approach and then by programming.
The reasons for explaining in this way were to show the differences between the PC-4 computer and ordinary calculators and also to get you thinking in terms of making programs.
The big difference between a computer and a calculator is that a computer can store the working procedure, the program.
By storing this program in the computer, the same repetitious tasks can be performed over and over with amazing speed.
On the other hand, with a calculator, the numeric inputs and functions must be entered again and again.
Take another look at Chapter 2 and you will realize how difficult this can be.
Of course, if a problem does not require much repetition, then the time it takes to develop a program may be excessive, and the work could best be done directly on a calculator.
The calculator is best suited for simple problems.
So the question as to which is better, the calculator or the computer becomes relevant in light of the individual job.
In any case, the PC-4 provides the convenience of a powerful computer with calculator capabilities.



BASIC is unexpectedly simple.

We think that you probably understand now how to use the programs which were practiced in Lists 1 through 5 of this chapter.
Also, you probably feel more comfortable with BASIC.
Certainly, the program commands which appeared in this chapter (IF ~ THEN, INPUT, PRINT, GOTO, etc.) and the commands used outside the program (LIST, CLEAR, RUN, etc.) are readily understood since they have generally the same meaning as they do in common usage.
With the exception of the = symbol, which means to substitute the right side result for the left, each symbol used in BASIC (+, <, ~) is similar to mathematical terms.
Therefore, you were probably able to cope with these terms and symbols without any particular difficulty.
By this time, if your understanding of computers and of BASIC programming has increased even slightly, you will be able to easily master the following Chapter.

Chapter 4

Now Let's Learn BASIC

4.1 BASIC and the PC-4

BASIC programming language was developed in the United States. BASIC is an abbreviation for "Beginners All-Purpose Symbolic Instruction Code". Keep in mind that BASIC is a computer language for beginners. Even though BASIC has been used for a number of years, many people are now using it for the first time. As a result of the recent boom in personal computers, the use of BASIC is rapidly spreading throughout the world.

And yet, technological progress never stops.
BASIC language has now even arrived to the pocket-sized computer level.
Although the PC-4 does not have a graphic function, it is comparable to the full-sized personal computers of just a few years ago.
In spite of its small memory, the PC-4 uses full-fledged BASIC and not an abbreviated form. For example, when a program is listed, the display will shown PRINT or INPUT without abbreviation.
Therefore, the BASIC programs which you learn using the PC-4 can be used with other personal computers with little or no change.
That is why it may be said that the PC-4 is an excellent computer for beginners to learn BASIC.
Now let's take a look at some practical uses of BASIC.

4.2 Commands required to perform addition, subtraction, multiplication and division

In the previous chapter, a program for obtaining data sum and square sum simultaneously was introduced.
In a similar manner, we will now see a method for obtaining the sum, difference, product and quotient of two values.

See the List examples on pages 64, 67 and 68.

- **Remember the equal symbol (=) is different from the mathematical equal sign.**

By the way, in PC-4 BASIC, 26 character variables from A to Z can be used. However, since each variable may only be one character long, variables such as A1, AB, etc. cannot be used.
The program for writing the sum, difference, product and quotient of X and Y into A, B, C and D is as follows.

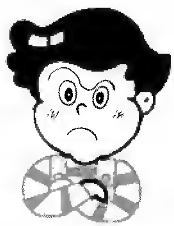
Commands which are used in a program are called "program commands". Command such as RUN or LIST which are not used in a program are called "processing commands".

List (a)

```
100  A=X+Y
110  B=X-Y
120  C=X*Y
130  D=X/Y
```

It should be noted here that the items to the left of the = symbol must be variables. In the above example, A, B, C and D are the variables.
Also, the items on the right side of the = symbol are called the "mathematical formulas" or simply "formulas". Of course, numerical values can also be placed on the right side.

Understanding the use of the = symbol is the beginning of understanding programming. Be careful since it is easy to misunderstand.



The = symbol means "substitute the value obtained using the formula on the right side for the variable on the left side".
Therefore, it has a meaning which is different from that used in mathematics. Be careful.

INPUT and PRINT commands

In order to execute List (a), it is necessary that both the value of X and the value of Y be given.
Then the obtained value must be displayed.
First, input the value of X and Y as follows.

```
10 INPUT X
20 INPUT Y
```

This can also be written on one line as follows.

List (b)

```
10 INPUT X, Y
```

When one line is used, be sure not to forget the " , ".
A "?" will be displayed for each variable to request an input.
It's very simple and convenient to be able to use one line, isn't it?
Next, to display the values of A, B, C and D, input the following.

List (c)

```
200 PRINT A
210 PRINT B
220 PRINT C
230 PRINT D
```

As with the INPUT command, this can be done using only one line:

```
200 PRINT A, B, C, D
```

Now let's put Lists (a), (b) and (c) together to make List 6.

List 6

```
 10 INPUT X, Y
100 A=X+Y
110 B=X-Y
120 C=X*Y
130 D=X/Y
200 PRINT A
210 PRINT B      These should be written on
220 PRINT C      one line as follows;
230 PRINT D      200 PRINT A, B, C, D
240 END ........ Terminates the program
```

END on line 24∅ means the program is complete. But the program can still be executed even without it.

● **Program write-in and execution.**

Now let's write the List 6 program into the PC-4 and execute it.

Put the PC-4 in the WRT mode in order to write in the program.

See page 62 of Chapter 3 for an explanation of the WRT mode and the RUN mode.

[MODE] [1] .... Designates the WRT mode.

[C] [L] [E] [A] [R] [A] [EXE]

After pressing the above keys, the display will be as follows and the ∅ will blink.

```
     DEG
   WRT
P   0 1 2 3 4 5 6 7 8 9
```

This means that there is no other program written in the PØ program area.

Then write in List 6.
Next, to execute the program, press the following keys to designate the RUN mode.

[MODE] [0]

The display will be as follows.

```
RUN
   DEG
READY P0
```

Then press the following keys.

[R] [U] [N] [EXE] or [S] [P0 Ø]

To execute a program other than PØ, for example the P1 program, press the following keys while in the RUN mode.

[S] [P1 1]

That program will be executed immediately.

Now let's execute the List 6 program which was written in program area PØ.
When execution begins, a "?" will be displayed. This is a request to input a value for X. Input the value.

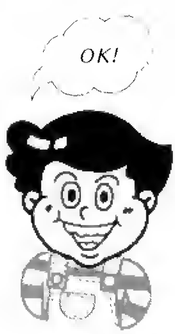
For example, input 3Ø. Then press the EXE Key again and another "?" will be displayed. This is a request to input a value for Y. For example, if you input 6Ø and press the EXE Key, 9Ø will be displayed. This indicates the value of A. At the time that this numerical value is displayed, STOP will appear on the upper right of the display.
This means that the program stops when it displays one answer. If EXE is pressed again, the value of B will be displayed and STOP will appear again.
The procedure and displays are as shown below.

Key	Display	
S P0 Ø	RUN DEG ?	Request to input value for X
3 Ø →	RUN DEG 3Ø_	Input 3Ø for X.
EXE →	RUN DEG ?	Request to input value for Y
6 Ø →	RUN DEG 6Ø_	Input 6Ø for Y. Awaiting next EXE key input
EXE →	RUN DEG STOP 9Ø	PRINT A execution (A = X + Y)
EXE →	RUN DEG STOP -3Ø	PRINT B execution (B = X − Y)
EXE →	RUN DEG STOP 18ØØ	PRINT C execution (C = X * Y)
EXE →	RUN DEG STOP Ø.5	PRINT D execution (D = X / Y)
EXE →	RUN DEG _	END execution

Character strings

The program in List 6 only uses two variables (X and Y). When more variables are used, however, it becomes unclear to which particular one the "?" applies. We will avoid this confusion by using a method known as character strings. These are easy to use. Simply enclose the character string in quotation marks (" ").
Character strings can be used in both INPUT statements and PRINT statements. For example, "ABC", "X=", etc.
Let's rewrite List 6 by adding character string.

It is important that the program be easy to understand. Let's use character strings to avoid confusion.



List 7

In case of an INPUT statement, insert a comma between the character string and the variable.

```
10  INPUT "X=", X
20  INPUT "Y=", Y
```

Lines 1Ø0 through 13Ø are the same as List 6

In case of a PRINT statement, insert a semicolon between the character string and the variable.

```
200  PRINT "SUM="; A
210  PRINT "DIFFERENCE="; B
220  PRINT "PRODUCT="; C
230  PRINT "QUOTIENT="; D
240  END
```

The command on Line 10 is to display the character string "X=" and to input a value for X. A comma (,) is required between "X=" and X.
The command on Line 200 is to display the character string "SUM=" and to display the value of variable A subsequently.
A semicolon (;) is required between "SUM=" and A. If the comma and semicolon are used incorrectly, an error will occur on lines 1Ø and 2Ø. On line 2ØØ, only "SUM=" would be displayed and subsequent pressing of [EXE] would show the sum in variable A. This also applies to lines 21Ø, 22Ø and 23Ø.

4.3 Repeating any number of times

FOR ~ NEXT loop

In Chapter 3, we made a program to obtain the total of the numbers 1 through 10 and the square sum. We also stated that the computer is excellent for performing such repetitious work.

In the program in Chapter 3, an IF ~ THEN conditional statement was used for input data. Here we will use a FOR ~ NEXT command to achieve the same thing.

First, let's look at an example.

List 8

```
10  A=0 : B=0
100 FOR I=1 TO 10
110 A=A+I
120 B=B+I↑2
130 NEXT I
140 PRINT "SUM TOTAL:" ; A
150 PRINT "SQUARE SUM:" ; B
160 END
```

The following statement appears on line number 100.

```
FOR I=1 TO 10
```

From now on, each collective command will be called a "statement".

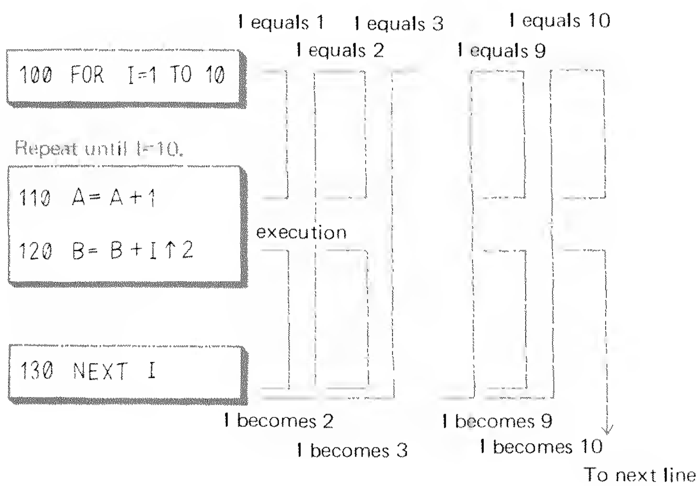
FOR means "while ________ ".

Then if "I = 1 to 10" is inserted above, this means

"while variable I takes values from 1 through 1∅, execute the operations on line numbers 11∅ and 12∅".
The following statement appears on line number 13∅.

NEXT I

This is a statement which corresponds to FOR. It means "increase the value of I by 1 only".
This kind of command is generally called a "FOR ~ NEXT loop".
In summary, the FOR ~ NEXT statement means "while variable I takes values from 1 to 1∅, repeat and execute the statements on the line numbers which are between the FOR and NEXT statements".
This operation is shown in the Figure below.



This kind of FOR ~ NEXT loop is used frequently in BASIC, so remember it well.
By the way, in this example, the variable was increased by 1. Depending on the situation, other numbers may be used as an increment. If 2 is used, for example, the increase in the variable will be something like 1, 3, 5, 7, etc.
To set a different increment other than 1, add STEP and the new number. In the case of increasing by 2, input the following.

```
FOR I=1 TO 11 STEP 2
```

If STEP is used, the variable will be increased in regular steps.

Using negative values, decrementing can also be done.
The variables used in FOR ~ NEXT statements are called "loop variables". The first value is called the "initial value". The last value is called the "final value". The incrementing value is simply called the "increase" or "step".

- **Multistatements and powers**

Please take another look at List 8. You will see two symbols which have not been used before.
First, if you look at line number 1∅, it reads as follows.

```
10  A=0 : B=0
```

A = ∅ is a statement to substitute ∅ for variable A.
B = ∅ is also a statement. However, there is a colon (:) separating the two statements.
In BASIC, it is possible to write more than one statement on a single line. This is called a "multistatement".
Next, if you look at line number 12∅, you will see the following.

```
120  B=B + I↑2
```

Power (↑)

The arrow pointing upward is the power symbol. For example, I ↑ 2 means I^2. If it were I ↑ 3, this would mean I^3, etc.

Any number may be used for the exponent value as long as it is a positive real number.
In the example below, write 2 as the value for variable A and then obtain the square, the third power, the square root and the one-third power.

	A=2	EXE	········ The cursor will blink.
A^2 ··················	A↑2	EXE	········ 4
A^3 ··················	A↑3	EXE	········ 8
$A^{\frac{1}{2}}$ (.5 is the same as 0.5)	A↑.5	EXE	········ 1.414213562
$A^{\frac{1}{3}}$ (parentheses required)	A↑(1/3)	EXE	········ 1.25992105

In the above example, a slight explanation is required about the one-third power. The 1/3 power is in parentheses because, as in mathematics, priority is given to operations enclosed in these.
Therefore, the operation to obtain the one-third power of A is written as follows.

A↑ (1/3) ·················· $A^{\frac{1}{3}}=\sqrt[3]{A}$

If the operation were written as shown below, without the parentheses, it would mean to "divide the 1st power of A by 3".

A↑1/3 ·················· $A^{1}/3$

● **Line insertion and changing**
Take another look at List 8. The initial value and final value in the loop variable are 1 and 1∅ respectively. These can also be changed to variables. List 9 shows changes made from List 8.

List 9

```
10  A=0 : B=0
20  INPUT "TOTAL=", N
100 FOR I=1 TO N
    :
    :   Line numbers 110 through 150
    :   are the same as List 8.
160 END
```

It is not necessary to write in all of the line numbers in order to test this program.
This is because the PC-4 has the following features.
(1) Even when the power switch is turned off, the stored program will not be erased.
(2) The line numbers are automatically arranged in ascending order. For example, even if line number 1ØØ is input before line number 1Ø, they will appear in the proper sequence when the program is listed.
(3) In case of inputs with the same line number, the last input cancels the previous input.
Suppose List 8 is written in program area PØ.
(a) Input line 2Ø statement as shown in List 9.
(b) Change line 1ØØ statement in List 8 as shown in List 9.
This will result in the old List 8 being changed to the new List 9.
As a test, input 1ØØØ for N and obtain the sum and square sum of the numbers from 1 to 1ØØØ.
Were the results as follows?

```
SUM TOTAL    : 500500
SQUARE SUM   : 333833500
```

Hey! Nothing's coming out! Be patient! The PC-4 is working very hard to do this difficult calculation. The answers will be displayed after about 2 minutes and 45 seconds.

4.4 Which one is the largest ?

When inputting data, we sometimes want to obtain the maximum value, the minimum value, and the average value.

This is especially true when we analyze test results or height and weight data.

To obtain an average value, total the data and divide by the number of data elements. You probably have an idea of what the program would look like.

But, how does the computer determine the maximum and minimum value, and how is the program written?

The approach for obtaining the maximum and the minimum values are basically the same. We will begin by explaining how to obtain the maximum value.

● **Program to obtain maximum value**

X is the variable used for data input. The variable for the maximum value is M.

First, input the least possible value for M.

For example, in the case of test results, there are no results below Ø so input M = Ø.

Next, compare M and X. If X is equal to or greater than M (X ≧ M), substitute the value of X for M.

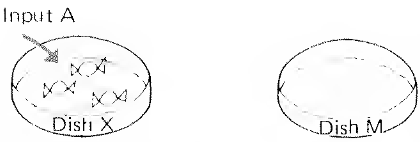
This only compares the number of data. If we express this approach using an illustration, it would look like the illustration on the next page.

There are 4 children, namely, A through D. These children have 3, 6, 4 and 8 pieces of candy respectively.
How many pieces of candy does the child with the most candy have.

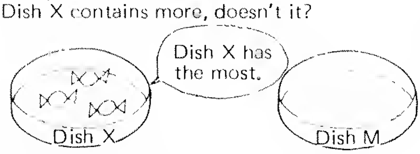
If we look at this problem using a computer approach . . .

(1) Prepare 2 empty dishes, Dish X and Dish M.

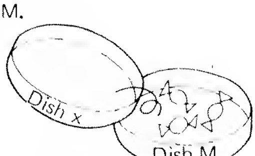
(2) Next, put all of child A's candy into Dish X.



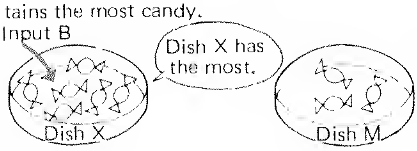
(3) Which dish contains the most candy, Dish X or Dish M?
Dish X contains more, doesn't it?



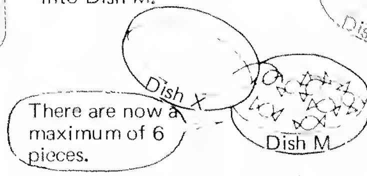
(4) Now move all of the candy from Dish X into Dish M.



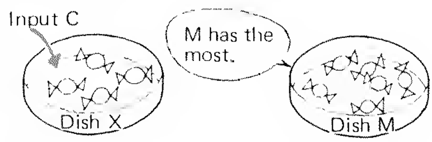
(5) Next, put all of child B's candy into Dish X.
Then compare to see which dish contains the most candy.



(6) Now take the candy from Dish M (which is the least amount) and put it aside. Then move all of the candy from Dish X into Dish M.

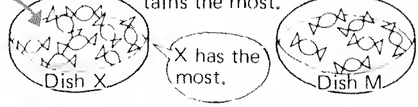


(7) Next, put all of child C's candy into Dish X. Then compare.

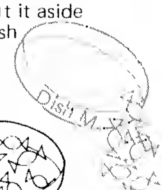


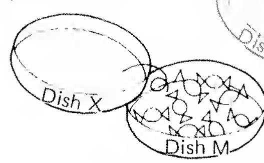
(8) Now take the candy from Dish X (which is the least amount) and put it aside. Then put all of child D's candy into Dish X. Now compare to see which dish contains the most.

Input D

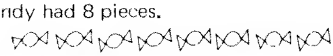


(9) Since M has the least, take the candy out of Dish M and put it aside and take the candy from Dish X and place it in Dish M.





(10) At this point, all of the candy has been placed into dish X, hasn't it? Well, how many pieces of candy remain in Dish M? 8, right?
This means the child with the most candy had 8 pieces.

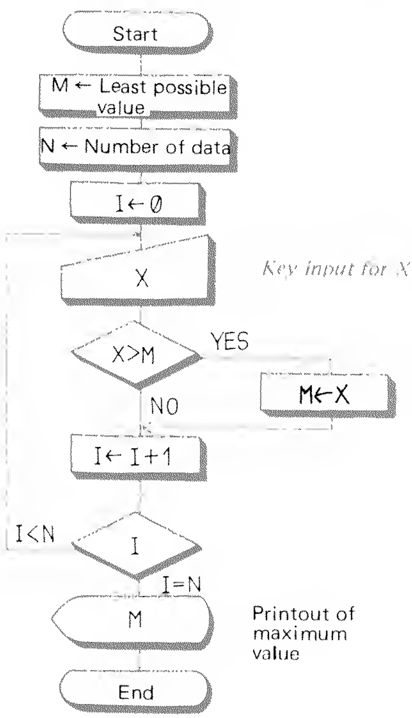


In this manner, the computer does not simply make one comparison but compares two at a time and repeats this a number of times to get the answer.

Do you have a rough idea of the computer approach to this kind of problem?
If we make the illustration of the previous page a little more formal, it will look like the figure below.
This figure is called a "flowchart". When programming, this is a very convenient tool because it makes the procedure very clear and easy to understand.
If we follow the flowchart we can make a program as follows.

```
10  M=0
20  INPUT "TOTAL=", N
30  I=0
40  INPUT "DATA=", X
50  I=I+1
60  IF X>M ; M=X
100 IF I<N THEN 40
110 PRINT "MAXIMUM=" ; M
140 END
```

Flowchart to determine maximum value



● IF and IF ~ THEN

Take a look at line 6∅ and line 1∅∅ in the above list.
The IF command is used differently on line 6∅ and line 1∅∅.
The IF command on line 6∅ means "if X is greater than or equal to M, execute the command written after the " ; " (semicolon) and substitute the value of X for M.
If this command is expressed grammatically, it will read as follows.

IF comparison formula; command

On the other hand, the IF command on line 1ØØ means "if I is less than N, jump to line 4Ø."
If this command is expressed grammatically, it will read as follows.

IF comparison formula THEN line number

In this manner, a command which begins with an IF can be used in two ways. This is very convenient when programming.

- **Obtaining minimum value and average value simultaneously.**

Now let's make a program to obtain the minimum value and average value simultaneously.
To obtain the average value, input the total of the data for variable A and divide by number of data N. Also, to obtain the minimum value, the approach is the opposite of that used to obtain the maximum value.
If S is already the variable for the minimum value, the program will be as follows.

List 10

Initially designate the highest possible minimum value as 1ØØ.

Initially designate the value for the data total as Ø.

```
10  M=0 : S=100 : A=0
20  INPUT "TOTAL=", N
30  I=0
40  INPUT "DATA=", X
50  I=I+1 : A=A+X
60  IF X<M THEN 80
70  M=X
80  IF X>S THEN 100
90  S=X
100 IF I<N THEN 40
110 PRINT "MAXIMUM=" ; M
120 PRINT "MINIMUM=" ; S
130 PRINT "AVERAGE=" ; A/N
140 END
```

Total is substituted for A at each data input.

If input data is greater than the current minimum value, go to 1ØØ. If equal to or less than, go to 9Ø.

Substitute data for S and make it the current minimum value.

Displays minimum value.

Divides the total by the number of data and displays that value. (This formula can also be written [illegible] PRINT.)

4.5 Challenging the troublesome deviation values

Recently, there has been an increase in the evaluation of grade school and high school tests using deviation values. You have probably encountered troublesome deviation values yourself.

By the way, do you know how to obtain deviation values?

The majority of the people probably do not know how to obtain it. Let's challenge this problem by using the PC-4.

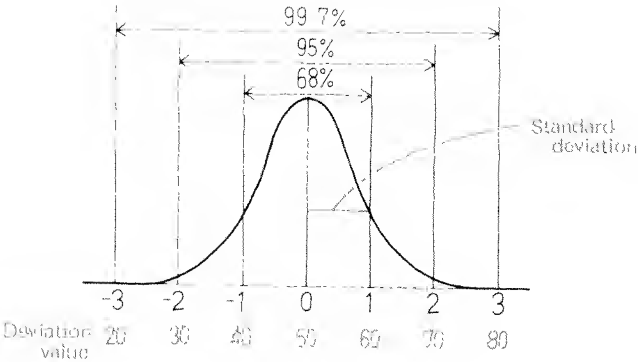
● What is the deviation value?

Let's assume that we are collecting test scores from a number of students and that we group similar scores together. Then let's assume that if we plot the number of students in each score range (this is called the "distribution"), as shown in the figure on the next page, a bell-shaped curve (this is known as the "normal distribution") is formed.

The point at the top of this bell-shaped curve is the mean of the distribution. Beginning there if we draw points such as ±1, ±2, ±3, etc., it can be seen that the number of students in the -1 to +1 score range is 68% of the entire group, that the number of students in the -2 to +2 range is 95% of the entire group, and that the number of students in the -3 to +3 range is 99.7%.

Then the deviation values are as follows: 50 at 0, 60 at 1, 70 at 2, 80 at 3, 40 at −1, 30 at −2 and 20 at −3.

It can be seen from the figure on the right that the students with a deviation value of 70 (that is 2 standard deviation intervals from the mean) are bright students and are in the top 2.5% of the class.



Only one or two students out of a thousand have a deviation value of 80. They are geniuses.
However, actual scores do not have this kind of normal distribution.

Changing the subject slightly, the complicated formula to obtain deviation values is shown below.
Looking at this formula, you may get a headache or be thoroughly confused. So just look at the part printed in blue.

Average value $= \sum_{i=1}^{N} x_i / N$ — Divide the total of the data by N

Standard deviation $= \sqrt{\left\{ \sum_{i=1}^{N} x_i^2 - N \times (\text{average value})^2 \right\} / (N-1)}$

Subtract the product of N times the average value squared from the square sum of the data and divide the result by N minus 1, then take the square root of that result.

Deviation value $= 50 + 10 \times (B - A)/C$

A: average value B: your score C: standard deviation

● Program to obtain deviation values

Before beginning programming, please take a look at List 4 on Page 67.

This was a program for obtaining the total of N numbers of data elements and the square sum simultaneously. If we use the same programming, we will be able to solve the problem smoothly.

For example, if we make A the variable for the total of N number of data, and make C the variable for the average value, the formula for obtaining average value C is as follows.

$$C = A/N \cdots\cdots (1)$$

Also, if we make B the variable for the square sum, value V is the square sum of the data minus the number of data times the average value squared.

The formula for obtaining value V is as follows.

$$V = B - N * C * C \cdots\cdots (2)$$

Then if the square root of the value divided by the number of data elements minus 1 is taken, this becomes the standard deviation D. The formula for obtaining the standard deviation D is:

$$D = \sqrt{V / (N - 1)} \cdots\cdots (3)$$

Then to obtain deviation value H when your score is Y, the formula is as follows.

$$H = 5\emptyset + 1\emptyset * (Y - C) / D \cdots\cdots (4)$$

In List 4, an IF ~ THEN statement was used to make a loop. However, at this time, we will make a loop using a FOR ~ NEXT statement.

Anyway, we will show you the list first.

List11

Repeats until N number of data has been input.

Endless loop

```
10  VAC
20  INPUT "TOTAL=", N
30  FOR I=1 TO N
40  INPUT "DATA=", X
50  A=A+X : B=B+X*X
60  NEXT I
70  C=A/N  ································· Formula (1)
80  V=B-N*C*C  ·························· Formula (2)
90  D=SQR (V/(N-1))  ···················· Formula (3)
100 PRINT "AVERAGE="; C
110 PRINT "STANDARD DEVIATION=" ; D
120 INPUT "SCORE=", Y
130 H=50+10*(Y-C)/D  ··················· Formula (4)
140 PRINT "DEVIATION VALUE=" ; H
150 GOTO 120
```

● **How to operate**

Input the following data and obtain the average value, the standard deviation and the deviation value.

Total: N = 1Ø

Data : 3Ø, 5Ø, 8Ø, 4Ø, 2Ø, 7Ø, 6Ø, 9Ø, 5Ø, 8Ø

If you execute this program (press [R] [U] [N] [EXE] or [S] [RUN/8] [EXE]), the PC-4 will request the total as follows.

TOTAL = ?

Input the number of data elements (1Ø) and press the [EXE] Key.

Then the PC-4 will request the data as follows.

DATA = ?

Input the ten numbers.
Then the following will be displayed.

AVERAGE =57
[EXE]
STANDARD DEVIATION =23.11805451

Then press the [EXE] Key and the PC-4 will request your score as follows.

SCORE = ?

If your score was 9∅, input this. Then the following will be displayed.

DEVIATION VALUE =64.27······

This program has no end since the last line is GOTO 12∅, which causes the computer to ask for the score repeatedly.

To stop this, press the [STOP] Key. (However the [STOP] Key only stops program execution. To completely finish the processing, just press [MODE] [∅] .)

● **VAC**

In PC-4 BASIC, be sure to use VAC independently on one line.

Take another look at List 11. There are some commands which have not been mentioned before.
One of these is the VAC on line 1∅.
VAC is an abbreviation for "variable all clear".
It clears all of the values for variables A through Z.
In this program, it was used to make variables A and B ZERO (∅). However, at first it may be better to write each step in the program (i.e. A = ∅, B = ∅) so that you can see what is actually happening.
Also as will be explained later, it may be troublesome to use VAC when calling up a subroutine.

● SQR and Parentheses

SQR, which appears on line 9∅, is a function to obtain the square root and means the same as "$\sqrt{\ }$".
(Also SQR works the same as the ↑.5 of the X↑.5 which was explained on page 82.) SQR is always followed by a variable or formula as shown below.


```
SQR (variable) or SQR (formula)
```

In PC-4 BASIC, the formula can be a single variable. For example,

```
SQR A
```

However, in the case of

```
SQR A/(N-1)
```

SQR A is executed first and then the result is divided by (N-1).
When you desire to obtain the square root of an entire formula, the input must be written as follows.

```
SQR (A/(N-1))
```

The entire formula must be enclosed in parentheses.
This appears a bit confusing at first glance. In mathematics, parentheses and brackets are used to separate operations as shown below.

```
[A/(N-1)]
```

However, programming languages, including BASIC, only use parentheses. Therefore, the sequence of the parentheses and the corresponding operations are very important.
For example, on line 90, the formula is written as follows.

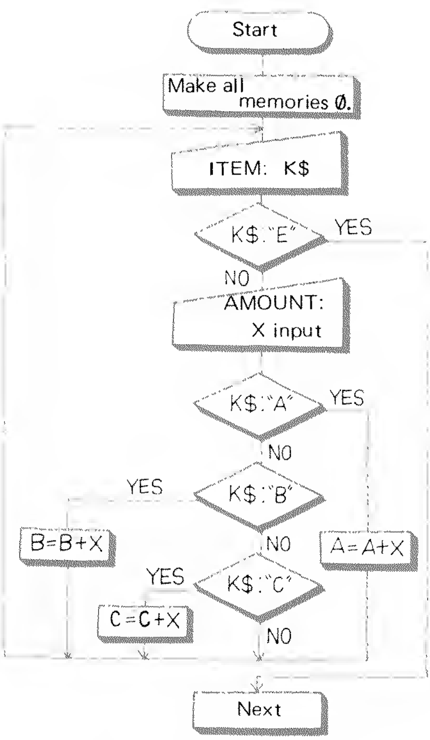
Same as mathematical parentheses

```
SQR (A/ (N-1))
```

Same as mathematical brackets

4.6 Obtaining separate totals

One of the many features of a computer is its ability to separate and group data.
For example, suppose a store wants to separate three categories of items (this is called "itemizing") and to calculate the total price for each item.
When this is made into a program, the main outline can be thought of as follows.



Flowchart for item classification

(1) Express the three items as A, B and C.

(2) First, input one of the items.

(3) Determine which item was input.

(4) Compute the item total.

(5) Input another item.

(6) Display the total for each item and the grand total.

Steps (1) through (5) in the above approach are explained in greater detail in the figure on the left.
Now let's take a look at the program list which corresponds to the figure.

List 12

```
10  A=0 : B=0 : C=0 : S=0
20  INPUT "ITEM", I$              Inputs the item
30  IF  I$="E" THEN 120           If E is input, the total compu-
                                  tation will be terminated.
40  INPUT "PRICE", X
50  IF  I$ ="A" THEN  90
60  IF  I$ ="B" THEN 100          Item separation
70  IF  I$ ="C" THEN 110
80  GOTO 20        (This will later be changed to read 80 PRINT "ITEM
                   ERROR!!" GOTO20) See below
90  A=A+X  : GOTO 20
100 B=B+X  : GOTO 20              Computes total for each item.
110 C=C+X  : GOTO 20
120 PRINT"A=" ; A
130 PRINT"B=" ; B                 Displays total for each item
140 PRINT"C=" ; C
150 S=A+B+C                       Computes grand total.
160 PRINT "GRAND TOTAL= "; S      Displays the grand total.
170 END
```

The example in List 12 is very simple in format. However, it illustrates a very convenient operation.

Character variables

In List 12, a new type of variable appeared, namely, I$.

We already know that there are 26 variables from A through Z. However, this is the first time that a variable with a symbol attached has been used.

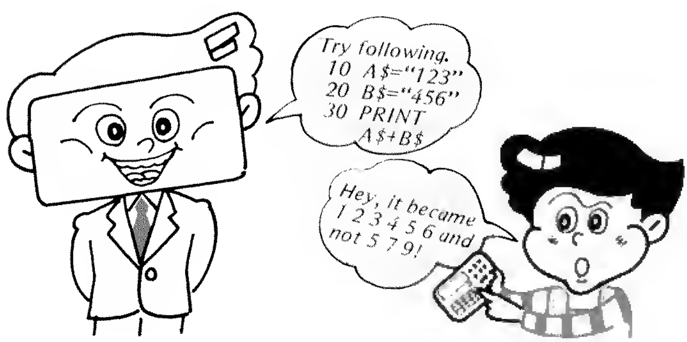
Variables with this symbol ($) attached are called "character variables" and character strings can be input into these variables.

This can be considered to be a variable version of character strings (characters enclosed by quotation marks) which have been previously explained.

The ability to use these character variables is one of the reasons why BASIC is used as a language for computers.

Line 20 means "input a value for I$". An alphabetical character must be input into I$ and not a number.

Numbers such as 123, 321, etc. could also be input but the computer will not interpret these numbers as numerical values. That is, they will not be interpreted as "one hundred and twenty three" or "three hundred and twenty one". Rather they are interpreted as separate characters such as "1", "2", "3" or "3", "2", "1". A maximum of 7 characters may be input into one character variable.



IF ~ THEN Precautions

IF ~ THEN have already been explained on page 86. However, there are some precautions that you should observe when using these commands.

Line 30 reads as follows.

IF I$="E" THEN 120

If this is written grammatically, it will read as follows.

IF conditional formula THEN line number

Therefore I$ = "E" is a conditional formula and not a substitution statement. Until now, we have seen that the equal symbol means "substitute the value on the right side for the variable on the left side". However, in the case of an IF ~ THEN statement, it is used differently. So please, be careful.
Here it means "if that conditional formula is properly concluded, then jump to line 120".

● **Program design**

If, in the program, there is a portion which is not consistent with the BASIC grammar, the PC-4 will indicate the error type and location. ERR2 P0-30, for example.

However, if the program design itself is faulty, problems could arise.
For example, suppose the List 12 program is executed (RUN) and "D" is input as an item.
Next, if a price is input, the program execution goes to line 20.
If you meant to input "A" but input "D" by mistake, the calculation to obtain the total for item A will not be executed, and consequently the total for A will be wrong.
Therefore, we must devise a system to prevent human error.
So we change line 80 of List 12 to read as follows.

```
80 PRINT "ITEM ERROR !!" : GOTO 20
```

Do this, input D, and see what happens.
The computer will tell you if you have input an incorrect item.
Of course, there are many prevention methods depending on the skill and desires of the programmer and the memory size the computer has.
Let's take a look at one more aspect of program design.

For the program List 12 on page 95, when the item and price are input, the computer requests the input of another item and price, continuing as an endless program. A method for terminating the program must be determined in order to read out the results.

Line 30 is used for this purpose.
If an "E" is input at this point, the program loop will be terminated and the totals will be displayed.

If "E" is then input when the PC-4 makes the request "ITEM ?", the totals for A, B and C and the GRAND TOTAL would be displayed and the execution terminated.

Note

Another character variable ($)
It has already been explained that in PC-4 BASIC, when a $ symbol is attached to the end of the respective variables A through Z, they become character variables. The length of the character string which can be stored in this character variable is limited to 7 characters. However, the $ symbol can be used alone as a character variable and permits storage of much longer character strings. In this manner, up to 30 characters can be stored.
As a test, select the RUN mode and input $ = "ABC...XYZABCD" EXE .
If you attempt to input 31 characters, ERR2 will be displayed.

4.7 Another type of variable —array variables

What are array variables?

We have already discussed variables in which numerical values can be stored and character variables in which characters can be stored.

There is another important variable that is used for programming. It is called an "array variable". Array variables are formed by attaching numbers enclosed in parentheses to the ends of variables A through Z (A(Ø), A(1), A(2) or B(Ø), B(1), B(2), etc.).

Array variables can store numerical values the same as variables A through Z.

It should be noted at this point that in PC-4 BASIC variable A and array variable A(Ø) operate in the same way. Correspondingly, array variable A(1) is the same as B.

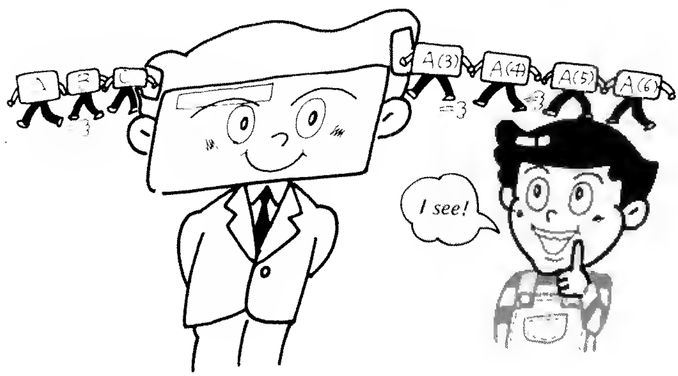
Since there are 26 letters in the alphabet, A(25) is the same as variable Z.

The table on the right shows the relationship of variables and array variables.

Table of relationships of variables and array variables

Variable	Array variables and variables which can be used at the same time.						
A	A(Ø)	A	A	A	…	A	A
B	A(1)	B(Ø)	B	B	…	B	B
C	A(2)	B(1)	C(Ø)	C	…	C	C
D	A(3)	B(2)	C(1)	D(Ø)	…	D	D
⋮	⋮	⋮	⋮	⋮		⋮	⋮
						X	X
Y	A(24)	B(23)	C(22)	D(21)	…	Y(Ø)	Y
Z	A(25)	B(24)	C(23)	D(22)	…	Y(1)	Z(Ø)

When you use an array variable, the equivalent single letter variable cannot be used. As can be seen from the table, the number of array variables and variables which can be combined is 26.



● **Why are array variables required?**

Since the number of variables which can be used is 26, you might wonder why having array variables makes any sense. The reason for having them is that they can help in simplifying programs that are otherwise complex.

For example, let's take another look at List 11.
This program was used to obtain the average score and standard deviation using the test scores of a certain number of students.
The basic procedure for this program is as follows.
(1) Input the scores of N students.
(2) Obtain the average score and standard deviation.
(3) Input the individual students' scores.
If we look at this program closely, the scores had to be input twice, once in Step(1) and again in Step (3). This is a waste of time.
Although you took great pains to input the students' scores in Step 1, this was of no use other than for storing them in the computer memory.

Let's assume that the number of students is 10 and you input the scores into the 10 variables from A through J as shown below.

```
INPUT A
INPUT B
   :        Input of the scores of the 10 students.
INPUT J
```

To make it easier to understand, let's input the following after the respective INPUT commands.

```
INPUT "A=", A
```

If done in this way, the program would be very long. So storing the scores in the memory is not such a good idea after all.
Besides, the capability of the computer to perform repetitious operations is not used to its full advantage.
This is where the array variable comes into play.
For example, let's consider the processing of the scores of the 10 students and use array variables A(0) through A(9). Remember that variables A through J cannot be used at this time.
First, let's key in the part of the program for inputting the scores of the 10 students.

Assign numbers from 0 through 9 to the students and input each score into an array variable.

List 3(a)

```
              10  N=10
              20  FOR K=0 TO N-1
Repeat 10     30  PRINT "K=" ; K ;
times         40  INPUT "DATA=", A(K)
              50  NEXT K
```

By simply doing this, the scores of the 10 students are input into variables A(0) through A(9).

Then we can easily obtain the average, standard deviation and deviation value.
At this time, since the scores of the students are input into memories A(0) through A(9) (actually A through J), the program is as follows.

List 13 b

```
100  P=0:Q=0        ······ P is the variable into which
                           the total and average are
                           input.
                           Q is the variable into which
                           the square sum is input.
110  FOR K=0 TO N-1
120  P=P+A(K)
130  Q=Q+A(K)↑2
140  NEXT K
150  P=P/N          ········ Average
160  V=Q-N*P*P
170  V=SQR(V/(N-1))  ···Standard deviation
```

Up to here, the average and standard deviation have been computed in preparation for obtaining the deviation values. The deviation values can be calculated in sequence for student numbers 0 through N-1. (In this case, 0 through (10-1), or 0 through 9.)

List 13 c

```
200  FOR K=0 TO N-1
210  X=50+10*(A(K)-P)/V  ···Deviation value
220  PRINT "DEVIATION VALUE:";K;"=";X
230  NEXT K
240  END
```

If Lists (a), (b) and (c) above are combined, you will get a single program. Using this program, you can store all of the scores and obtain the deviation values of all of the students.
By using array variables, the scores need not be input twice.

Let's become skilled at using "program-like" variables.

In the previous program, the variables on lines 120, 130, 150 and 170, etc., are used in a manner which is unthought of in mathematics.

This is because in BASIC, as was explained previously, the equal symbol (=) is used differently than the equal sign used in mathematics.

For example, lines 160 and 170 read as follows.

```
160  V=Q-N*P*P
170  V=SQR(V/(N-1))
```

The use of V here seems confusing.

The contents of the blue Vs on line 160 and line 170 are the same. However, the contents of the black V and blue V will be different because the value computed using the blue V is substituted for the black V.

Also, if it is not necessary to retain the scores of the individual students and only the deviation values are desired, line 210 may be written as follows.

```
210  A(K)=50+10*(A(K)-P)/V
```

In this manner, if you become skilled at using the same variables over and over, you will avoid the confusion of using numerous variables.

Note

◇ Expansion of Variables (DEFM) ◇

Normally, 26 variables (from A through Z) are used with the PC-4.

When array variables such as A(0) through A(25) are used, the number of variables is still 26.

If you press [S] [V] (DEFM) [EXE], the display will show ***VAR:26. This indicates that 26 variables can be used.

If you want to use more than 26 variables, press [S] [V] (DEFM), enter a numerical value, then press [EXE]. In this manner, the number of variables which can be used will be increased by the number which was input as the numerical value.

For example, if 5 is input as the numerical value, array variables A(26) through A(30) can also be used. ***VAR:31 will be displayed.

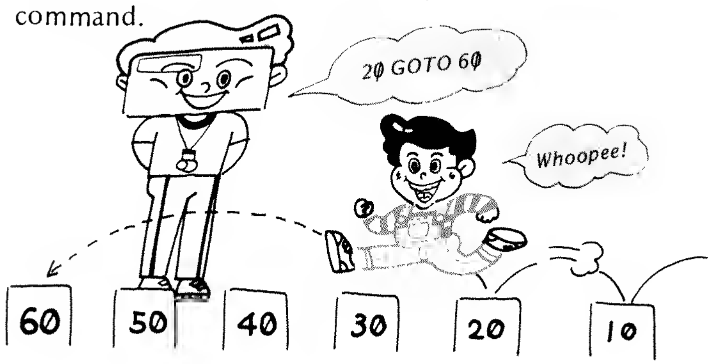
A maximum of 94 variables can be used.

4.8 Jump ! —GOTO

It has already been explained that the PC-4 reads commands in line number sequence from low line numbers to high line numbers.
However, regardless of the line number, it is possible to jump forward and backward.
The command to accomplish this is the GOTO statement.

● How to use the GOTO statement

As mentioned on page 96, an IF ~ THEN statement can also be used to jump forward and backward.
However, the IF ~ THEN statement causes a jump only if a certain condition is met. The GOTO statement jumps to the desired line number without any condition being set.
The GOTO command is always accompanied by a line number. The line number indicates the location where program execution will jump. GOTO 100, for example, causes program execution to continue in line number 100.
However, this is not the only use of the GOTO command.



Variables or formulas can also be used as destinations for a GOTO command.
Of course, there must be a line number to identify where the variable or formula is located. The jump destination can be changed, however, if the value of the variable or formula in question also changes.
For example, suppose we have a program as follows.

```
100  GOTO A
200  S=X+Y
300  P=X*Y
```

If the value of A is 200, the sum of X and Y will be obtained. If the value of A is 300, the product of X and Y will be obtained.
The above example is simple but, nevertheless, it gives an idea of the programming concept.

List 14

```
 10  INPUT "SUM:1  PRODUCT:2", B
 20  IF B=1 THEN 50
 30  IF B=2 THEN 50
 40  GOTO 10
 50  INPUT "DATA X=", X
 60  INPUT "DATA Y=", Y
100  GOTO B*100+100
200  S=X+Y
210  PRINT "X+Y=" ; S
220  END
300  P=X*Y
310  PRINT "X*Y=" ; P
320  END
```

When B is other than 1 or 2

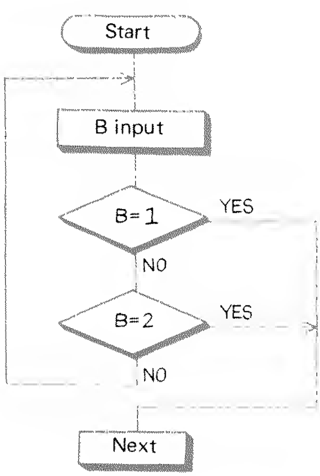
When B is 1

When B is 2

In this program, if 1 is input into variable B, the value obtained from the formula in the statement GOTO B*100+100 will be 200(1×100+100) so the program will jump to line 200 and obtain the sum of X+Y.
If 2 is input into variable B, the value obtained from the formula will be 300(2×100+100) and the program will jump to line 300 and obtain the product of X and Y.
If any other number than 1 or 2 is entered, the program returns to line 10.

● The program compensates for human error

The portion of the program from line 10 to line 40 is a programming method to compensate for human error and one that's used frequently in actual programming.
If we express this using a flowchart, it will appear as shown in the figure below.
For example, if 0 or a value of 3 or greater is input into variable B, the GOTO command on line 100 cannot be executed. As soon as execution is attempted, an error will occur.



The program accomplishes this by determining if the value input for B is correct (either 1 or 2).
If it is not correct, the program will recycle and make notification that the value input was other than 1 or 2.
Line 40 performs this operation. You should remember this use of the GOTO statement since it is a fundamental programming technique.

4.9 Frequently-made errors

So far, we have written and executed various programs. While doing this, you probably encountered the following kind of display quite a few times.

ERR 2 PØ−3Ø

In BASIC, various kinds of errors can occur (grammatical errors, mathematical errors, etc.).
The PC-4 can recognize nine kinds of errors, as shown in the instruction manual. Notification of these errors is made using ERR 1 through ERR 9 on the display.
You should be able to understand the meaning of each of these errors so that you know what is happening when they occur. At this time, we will take a look at a few of the most commonly made errors.

Even though you write in a manner which is easy for a human being to understand, the computer may not accept it.

[Example 1]

In BASIC language, the grammar permitted to be used for writing a program has been predetermined. If the rules of this grammar are violated in the slightest way, an error will occur.
For example, suppose you attempt to make the following input.

1Ø IF X=1 THEN GOTO 1ØØ

If GOTO is input immediately following THEN, a grammatical error occurs (100 should follow THEN and GOTO should be deleted). The display will show:

ERR2 PØ−1Ø

ERR 2 means a grammatical or syntax error has been made.
PØ-1Ø means the error is on line 1Ø in program area PØ.

[Example 2]

Next, we will discuss mathematical errors.

Instead of using a program, we will use examples in the RUN mode (manual calculation).

For instance, suppose we attempt the following.

[X][=][0][EXE]

[Y][=][1][0][/][X][EXE]

This will cause an error and the display will show

```
ERR3
```

The calculation attempted, 10÷0, is not mathematically possible. ERR3 indicates that a mathematical error has occurred.

Let's try another one.

[S][Q][R][−][1][EXE]

This is an attempt to obtain the square root of −1 ($\sqrt{-1}$). This is known as an "imaginary number". Therefore, a result cannot be obtained and ERR 3 will occur.

[Example 3]

Another kind of error may occur when a GOTO statement or a GOSUB statement (GOSUB will be explained later) has a destination line number which is not contained in the program.

Let's attempt to input and RUN the following.

```
10 GOTO 20
```

If the program does not contain a line 20, an error will occur and the display will show.

```
ERR4 P0-10
```

Note

◇ MID function ◇

The Note on page 98 discussed character variables. The MID function is used to extract a portion of the character string of the character variable. The MID function has two arguments and is written as MID (N, M). This means "extract M number of characters beginning with the Nth number from the left of the character string. Let's execute the following.

```
10 $="SMITH BROWN JONES"
20 PRINT MID(7, 5)
```

This will result in the name BROWN being extracted.

4.10 The game's leading players -random numbers

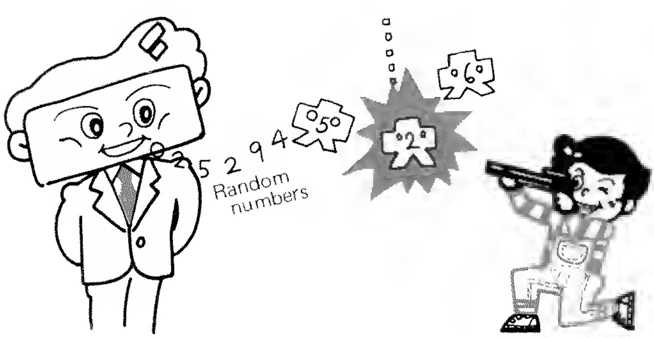
The story of random numbers

Have you ever played a video game? Random numbers play a leading role in video game programs.
For instance, consider the invader games. You never know where the invaders will appear or where they will attack. They appear and disappear at random and quite suddenly.
This unpredictability is one of the most interesting feature of video games and one reason why people enjoy them so much. The actual cause of all this appearing and disappearing is due to random numbers.
Let's take a more detailed look at random numbers.

For example, suppose we have the following series of numbers.

1 2 3 4 5 ☐ 7

Can you guess what number should be in the square? That's right. It's 6.



Well, how about the following?

1 4 9 16 ☐ 36

This is a bit more difficult but you can see that the series progresses by the squares of the numbers 1 through 6.
Therefore, the number in the square should be 25.
How about the following.

74 40 52 62 86 8 80

This time, the number is already shown in the square. However, probably none would guess why the number is 8. Even if you guessed it,this doesn't mean you are a genius because in this case it is just a random number. Any number could have been used.
The numbers in this kind of unrelated series of numbers are called random numbers.
In other words, random numbers have no set pattern so there is no way to predict the next number in advance.

● Function for generating random numbers – RAN#
The PC-4 has a function for generating random numbers. It is

RAN#

Select the RUN mode and press the following keys.

[R] [A] [N] [S] [#/F] [EXE]

If you execute this function several times, you will notice that a different series of numbers appears on each successive display.
The number is a decimal value between 0 and 1. However, there is no way to predict what number will appear next.

Random number generation and dispersion

Theoretically, a random number has an equal chance of being selected as any other number. To verify this, we could actually check the dispersion of random numbers generated 100 times with the following program.

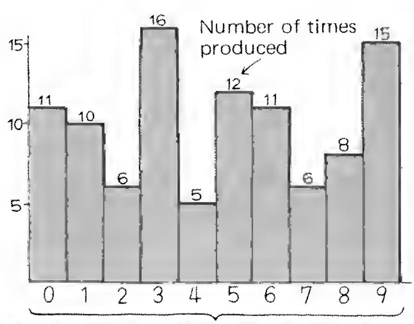
List 15

```
10 FOR I=1 TO 100
20 PRINT RAN#*10
30 NEXT I
40 END
```

Generate random numbers

Repeat 100 times

If we were to multiply each random number generated by 10, drop the decimal portion, and plot the frequency of each number, the graph would look like this.



As you can see, there is no set pattern. If the number of random number generations were to increase, the dispersion would become more even.

For example, if we generated random numbers between 0 and 1 repeatedly toward infinity (this is not actually practical) and divided the sum of all of the random numbers by the number of generations, theoretically, the average would be 0.5 and the standard deviation would be $\sqrt{1/12} = 0.288.....$

4.11 The ABC's of games

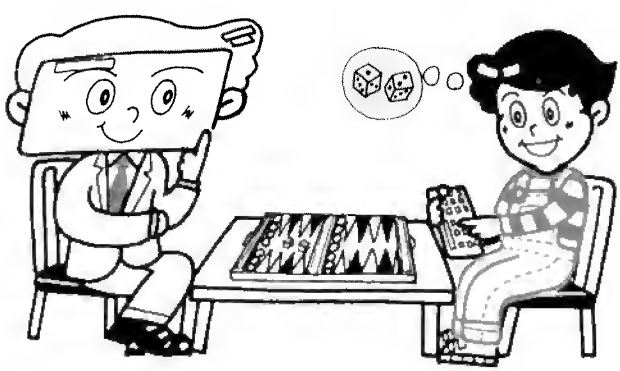
As was demonstrated in the previous discussion, a balanced set of meaningless numbers will be obtained. It was also said that random numbers play a leading role in many games.

● Electronic dice

In a game where dice are used such as backgammon, when the dice are thrown (assuming that there is no cheating) random numbers from 1 through 6 will come up without any particular pattern for each die. In this manner, since the numbers come up randomly, we are able to play the game.

Let's use the RAN# function and play a dice game using the PC-4.

Let's call the game "electronic dice". The program for "electronic dice" is as follows.



List 16

```
10  PRINT "ELECTRONIC DICE"
20  X=INT (RAN# *10) +1
30  IF X>6 THEN 20
40  PRINT "PRESENT NUMBER:"; X
50  GOTO 20
```

Multiplies the random number by 10, makes it an integer, adds 1 to it and substitutes it for X.

Let's write this program into the PC-4 and execute it. The first time the keys [S] [RUN B] [EXE] are pressed, the words "ELECTRONIC DICE" will be displayed. Then, press the [EXE] Key and the following will be displayed.

PRESENT NUMBER : number (1 through 6)

This completes "electronic dice".

INT function

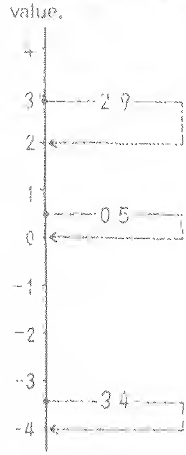
In the above program, a new command was introduced, namely, INT.
This is a function to change a decimal to an integer.
This is expressed grammatically as follows.

INT numerical value or INT (formula)

Then, the highest integer not exceeding the given numerical value will be obtained.
For example,

INT 2.9 ⟶ 2   INT 10.6 ⟶ 10
INT 0.5 ⟶ 0   INT −3.4 ⟶ −4

INT (numerical value) is rounded off to an integer having the least possible value.



Line 20 in the List 16 program generates a random number using the RAN# function (for example, 0.5701793564), multiplies it by 10 (5.701793564), makes it an integer using the INT function (5), adds 1 to it (6) and substitutes it for X.
Since the numbers on a die are only 1 through 6, when the number selected exceeds 6, line 30 causes the program to return to line 20 in order to reselect the number.

4.12 Dice game

In the previous section, we made an electronic dice program.
Let's change this program slightly and make a simple dice game.
You and another player will roll the die one time each.
The one getting the larger number is the winner.
First, let's take a look at the program.

List 17

```
10  PRINT "DICE GAME"
20  GOSUB 200
30  Y=X
40  PRINT "YOUR TURN :"; Y
50  GOSUB 200
60  Z=X
70  PRINT "MY TURN :"; Z
80  IF Y>Z THEN 120
90  IF Y=Z THEN 140
100 PRINT "I WIN"
110 GOTO 10
120 PRINT "YOU WIN"
130 GOTO 10
140 PRINT "TIE"
150 GOTO 10
200 X=INT (10*RAN#) +1
210 IF X>6 THEN 200
220 RETURN
```

Compares your number and the other player's.

Main routine

Subroutine

First jump to subroutine

First return

Second jump to subroutine

Second return

GOSUB and RETURN

In List 17, new commands were introduced, namely, GOSUB and RETURN.

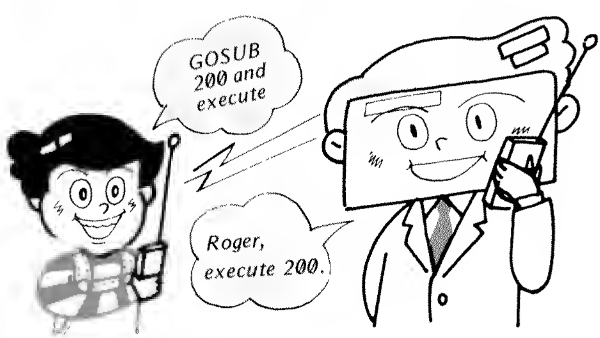
These commands mean "jump to the line number designated in the GOSUB command, execute the commands of the designated line number and all subsequent line numbers until a RETURN command appears, then return to the line following the original GOSUB command".

For example, for the dice game in List 17, the number on the die must be obtained each turn using the "electronic dice" program from the previous section. This means that the part of the program which performs the same operation would appear twice in one program.

This is a waste of time, so we write the program as shown using lines 200 through 220 and remove the operations from the main program flow and use a GOSUB command when these operations are required.

A program of this type which is outside of the main program flow is called a "subroutine". The main program is called the "main routine".

In this program example, the merits of the subroutine are not too apparent. However, in the case of a complicated program, the program can often be made much easier to follow by using a subroutine.

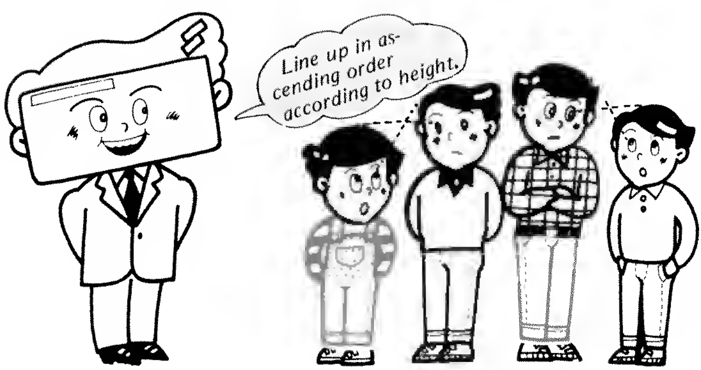


4.13 Line up !

In Section 4.4, we discussed obtaining the maximum and minimum values from among a number of data elements.
At this time, let's make a program which will not only select one item but will line up all of the data in ascending or descending order.
This ordering of the data is called "sorting".
We will discuss a method of sorting known as "bubble sort".

The sorting concept

As we discussed in Section 4.4, the computer cannot determine at a glance which number has the maximum value or which has the minimum value.
It will compare the data one by one and make a determination in sequence.
First, let's take a look at the sorting program only.



List 18-a Sorting subroutine

```
200 V=0 ........................ Flag
210 FOR X=0 TO 3
220 IF A(X)≦A(X+1) THEN 240    If the left side is larger, sort
                               on the next line.
230 Y=A(X) : A(X)=A(X+1) : A(X+1)=Y : V=1
240 NEXT X
250 IF V=1 THEN 200 ........... If V equals 1, repeat once
                               more.
260 RETURN
```

This is a program to sort five data elements into ascending order. Since there is a RETURN command on line 260, this program is handled as a subroutine.

The commands used have all been seen before. However, lines 200 and 230 introduce a new technique where the V is used as a flag, a kind of signal.

Let's take the five data elements shown below and perform the process that the computer uses.

A(0)	A(1)	A(2)	A(3)	A(4)
96	15	62	72	7

First, let's take a look at the sorting program!

Note

◇ SET Function ◇

The PC-4 displays numerical values as real numbers. To test this, suppose we obtain SIN 45°. The display will show 0.7071067812. It is good to have a large number, but sometimes only one decimal position is required and at other times you may want an exponential display. For this purpose, we use a function command called "SET".

SET is used as follows.

(1) SET N. . . . . Ordinary usage
(2) SET Fn (real number). . . . . n is a number from 0 through 9
(3) SET En (exponential number) . . . n is a number from 0 through 9

For example, in the case of (2), if SIN 45° is obtained using SET F5, the result will be displayed as 0.70711 with only five decimal places displayed. If this is performed using SET E5 the result will be displayed as 7.0711 E-01. This means "7.0711 x 10^{-1}".

Furthermore, the V on line 200 and line 230 is called a "flag". It operates as a kind of signal. Now let's use the data and follow the operation from line 200 through line 250.

First time Initial line up . . . A(0)=96 A(1)=15 A(2)=62 A(3)=72 A(4)=7

A(0)=96≤A(1)=15 (NO, to line 230) Y=96: A(0)=15: A(1)=96: V=1
A(1)=96≤A(2)=62 (NO, to line 230) Y=96: A(1)=62: A(2)=96: V=1
A(2)=96≤A(3)=72 (NO, to line 230) Y=96: A(2)=72: A(3)=96: V=1
A(3)=96≤A(4)=7 (NO, to line 230) Y=96: A(3)=7 : A(4)=96: V=1

Since V=1, jump to line 200 and repeat once more.

Second time Results of the first time . . . A(0)=15 A(1)=62 A(2)=72 A(3)=7 A(4)=96

A(0)=15≤A(1)=62 (YES, to line 240) . . . no sort : V=0
A(1)=62≤A(2)=72 (YES, to line 240) . . . no sort : V=0
A(2)=72≤A(3)=7 (NO, to line 230) Y=72: A(2)=7 : A(3)=72: V=1
A(3)=72≤A(4)=96 (YES, to line 240) . . . no sort : V=1

Since V=1, jump to line 200 and repeat once more.

Third time Results of the second time . . . A(0)=15 A(1)=62 A(2)=7 A(3)=72 A(4)=96

A(0)=15≤A(1)=62 (YES, to line 240) . . . no sort : V=0
A(1)=62≤A(2)=7 (NO, to line 230) Y=62: A(1)=7 : A(2)=62: V=1
A(2)=7 ≤A(3)=72 (YES, to line 240) . . . no sort : V=1
A(3)=72≤A(4)=96 (YES, to line 240) . . . no sort : V=1

Since V=1, jump to line 200 and repeat once more.

Fourth time Results of the third time . . . A(0)=15 A(1)=7 A(2)=62 A(3)=72 A(4)=96

A(0)=15≤A(1)=7 (NO, to line 230) Y=15: A(0)=7 : A(1)=15: V=1
A(1)=15≤A(2)=62 (YES, to line 240) . . . no sort : V=1
A(2)=62≤A(3)=72 (YES, to line 240) . . . no sort : V=1
A(3)=72≤A(4)=96 (YES, to line 240) . . . no sort : V=1

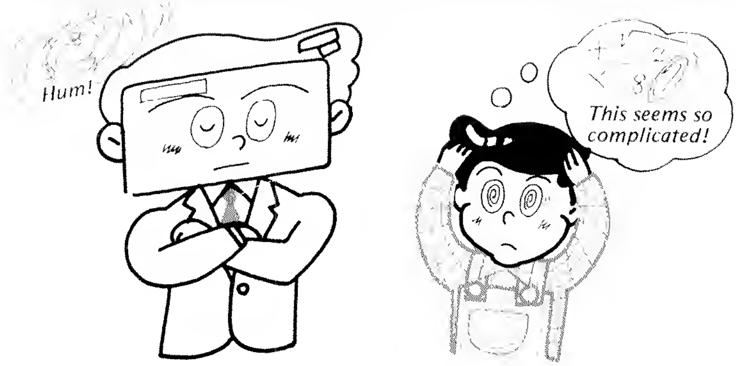
Since V=1, jump to line 200 and repeat once more.

Fifth time Results of the fourth time . . . A(0)=7 A(1)=15 A(2)=62 A(3)=72 A(4)=96

A(0)=7 ≤A(1)=15 (YES, to line 240) . . . no sort : V=0
A(1)=15≤A(2)=62 (YES, to line 240) . . . no sort : V=0
A(2)=62≤A(3)=72 (YES, to line 240) . . . no sort : V=0
A(3)=72≤A(4)=96 (YES, to line 240) . . . no sort : V=0

Since V=0, proceed to line 260 and sorting is complete.

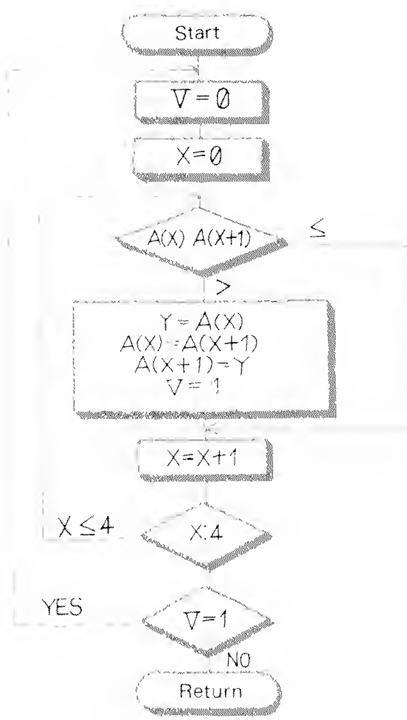
Results of the fifth time . . . A(0)=7 A(1)=15 A(2)=62 A(3)=72 A(4)=96



This seemingly complex and troublesome process is accomplished by the computer very quickly.
Did you understand the role of the V flag? It is very beneficial, isn't it? The flowchart for this sorting procedure is shown in the figure on the right.

In List 18(a), the sorting sequence was in ascending order. To sort in descending order, change line 220 to read as follows.

```
220 IF A(X)≥A(X+1)
       THEN 240
```



Program to input the data

When List 18(a) was explained, we presumed that the data had already been input, but actually the data had not yet been input so we have to make a program to input the data.

List18b Data input subroutine

Data is input 5 times.

```
100 FOR X=0 TO 4
110 INPUT "DATA=".A(X)
120 NEXT X
130 RETURN
```

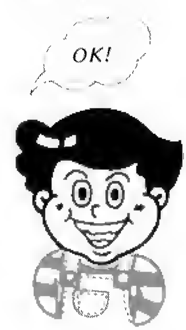
Return

This program is also considered to be a subroutine, so there is a RETURN command on line 130.
There are no particularly difficult statements on the other lines, are there?
This program is for requesting input of the five data elements from A(0) through A(4).

Subroutine to display the results

We have input the data and sorted it. Now we have to display the final results.
This program is also considered a subroutine. The program is as follows.

Here's a program to display the final results.



List18c Display subroutine

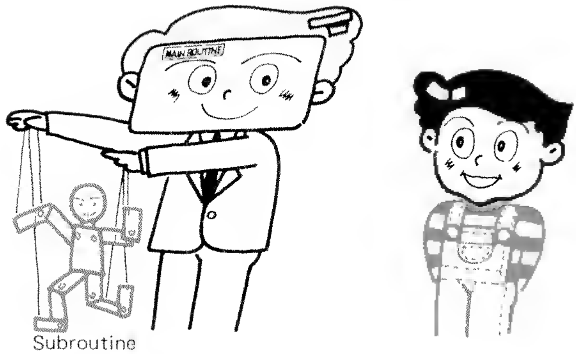
Display the results for the 5 data

```
300 FOR X=0 TO 4
310 PRINT X ; ":" ; A(X)
320 NEXT X
330 RETURN
```

Return

Main routine

So far, in List (a) through List (c), we made separate programs as subroutines.
However, these programs were simply executed in ascending line number sequence and there was no destination line number for the RETURN command.



Therefore, a main routine program is necessary to designate the subroutine return or to indicate which subroutine to go to from the main routine.

Finally, the main routine.

List18d Main routine

```
50  GOSUB 100      → to data input subroutine
60  GOSUB 200      → to sort subroutine
70  GOSUB 300      → to display subroutine
80  END
```

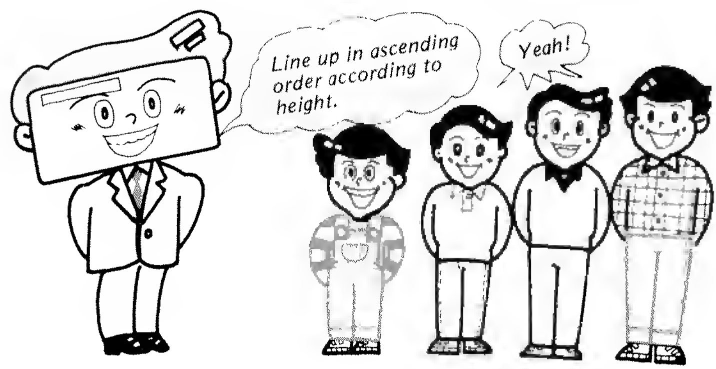
Programs 18(a) through 18(d) input the five data elements, sort them and display the results. However, if data is written in and the program is executed a number of times, the previous data will remain and the numerical values may change. In order to correct this situation, variables A(0) through A(4) must be initially cleared.

List18e

```
10  PRINT "SORT"
20  FOR X=0 TO 4
30  A(X)=0
40  NEXT X
```

Clears variables A(0) through A(4)

Now the program is complete.
Try inputting several different groups of data with five numerical values each.
If you would like to input and sort more data, simply change the "4" of the portion in each subroutine which reads "FOR X =0 TO 4" to correspond to the number of data elements you want.
The sort program discussed here is not suitable when the number of data is more than 100 elements because it takes a lot of time to run.
However, when there are not too many data elements, it is a good method to use.



Note

◇ Display of small letters or symbols ◇

BASIC is written using English capital letters.
However, the PC-4 has a function which allows characters, symbols or small English letters to be freely used in the contents of character variables in PRINT statements.
To use this special function, press MODE · . "EXT" (extension) will appear on the upper left of the display. In this mode, if the letters A through Z are pressed, small letters will be displayed. In addition, in this mode, if A through Z are pressed after the S Key, symbols such as Σ, Ω, ♦, ♣, etc. will be displayed. Try these.
To release the EXT mode, just press MODE · again.

4.14 Sorting of names as well

In the previous section, we performed data (number) sorting.
However, it would be a problem if we could only sort numerical values.
For example, suppose we would like to sort test scores. We would not know who had the highest score, lowest score, etc.
What? Test scores again?
You may get nervous when we talk about tests. However, this is just an example, so please be patient. To continue the discussion, the names of the students and the test scores would have to be sorted together to avoid confusion.
Let's make a program which can sort the names as well.

This program can sort names as well.

● Sorting of names

The subroutine for name sorting can be made by slightly adding to the subroutine (List 18(a)) in the previous section.
If we make the variable for inputting the names K$(X); the subroutine will be as follows.

List 19(a) Subroutine for sorting data and names

```
         200 V=0
         210 FOR X=0 TO 3
         220 IF A(X)<=A(X+1) THEN 240
         230 Y=A(X):A(X)=A(X+1):A(X+1)=Y:V=1
Addition 235 W$=K$(X):K$(X)=K$(X+1):K$(X+1)=W$
         240 NEXT X
```

```
250 IF V=1 THEN 200
260 RETURN
```

Line 235 is a statement for sorting the names.
One line 22∅, if the left side is greater than the right side, the data will be sorted on line 23∅ and the names will be sorted on line 235.

● Subroutine for inputting the data and names
This subroutine can also be made by adding one line to the subroutine (List 18(b)) of the previous section.

List19(b)

```
100 FOR X=0 TO 4
105 PRINT X , ": NAME=" ; : INPUT
    K$(X)
110 INPUT "DATA=", A(X)
120 NEXT X
130 RETURN
```

The names can be input as a result of adding line 105.

● Subroutine to display the results
This subroutine can also be made by adding a portion to List 18(c).

Just a slight change.

List19(c)

```
300 FOR X=0 TO 4          Addition
310 PRINT X ; ":" ; K$(X); "="; A(X)
320 NEXT X
330 RETURN
```

(Sequence):(name)=(score) is displayed as a result of line 310.

Main routine

The main routine can also be made by making a slight addition to List 18(d)/(e).

List 19(d)

```
10  PRINT "SORT"
20  FOR X=0 TO 4
                         Null
30  A(X)=0 : K$(X)=" "
                       Addition
40  NEXT X
50  GOSUB 100         Input of names and data
60  GOSUB 200         Sort of names and data
70  GOSUB 300         Display of results
80  END
```

On line 30, variable A(X) and character variable K$(X) are cleared. However, for character variables, characters are cleared by enclosing a space in quotation marks (this is called "null").

Null

Let's trim it up a little.

If we use this null, we can make a command as follows.

```
65  Z$=KEY : IF Z$=" " THEN 65
```

KEY function

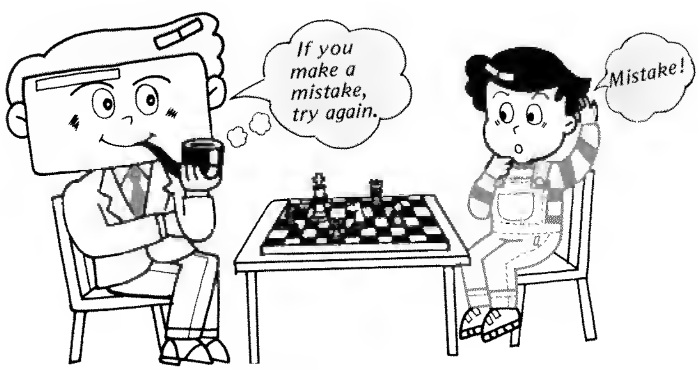
KEY means "the function which was input at that time using the keys". If no keys are pressed, nothing will be input for Z$, so the IF statement causes the program to return to line 65. Since line 65 is the same line number, returning results in execution being stopped at this line.

For example, if this line 65 statement is added to List 19(d), even though the execution of the sort subrou-

tine (lines 200 through 260) is complete, the display subroutine (lines 300 through 330) will not be executed immediately and the program will wait for some key input.
This is not really needed in this program. However, it prevents accidentally missing a display of the result.
Next, if the following statements are added, after input of each set of data and name, the computer will ask if the input is O.K. by displaying "(Y/N)". Y means "Yes" and N means "No".

```
55 INPUT "(Y/N)", U$
56 IF U$="N" THEN 10
```

This will help prevent the input of erroneous data and avoid useless results.
Checking for erroneous key input is an important part of programming.



This completes the program. Let's check it by inputting the following data and executing the program.

Finally, program execution!

FOX	50
WISE	80
FAGAN	40
HUNTER	60
STANLEY	70

4.15 Character variables also have sizes.

Character variables have a $ symbol attached to the variable such as A$, B$, etc. This, as you may recall, has been mentioned before.
Also, remember that up to 7 characters can be input into one character variable.
When the $ symbol is used alone as a character variable, up to 30 characters can be input.
These were explained in the Note on page 98. Character variables can be used as shown below by enclosing the character strings in quotation marks and substituting for A$ and B$.

```
A$="FOX"
B$="WISE"
```

The reason for explaining something you already know is that the size of the contents of character variables such as A$, B$, etc. can also be compared to numerical variables.

Program to compare the size of character strings.
If it were possible to put the names of people or products into character variables and sort these in sequence by comparing their size, this would be quite convenient, wouldn't it?
Anyway, actions are more important than words, so let's try it out.
Write the following program into the PC-4.

List20

```
10 INPUT "DATA1S", AS
20 INPUT "DATA2S", BS
30 IF AS>BS THEN 60
40 PRINT AS;" < "; BS
50 GOTO 10                    Space
60 PRINT AS;" > "; BS
70 GOTO 10
```

If this program is executed, the computer will display

DATA 1$?

Input a name, for example, "FOX".
Next, the following display will be.

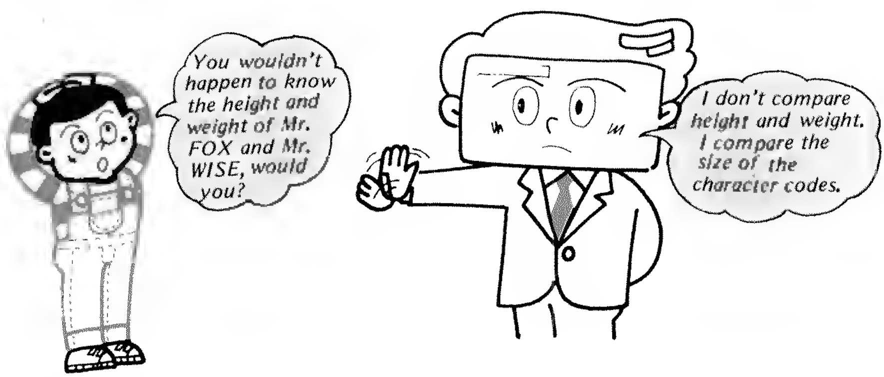
DATA 2$?

Input "WISE".
Then, the display will be as follows.

FOX≤WISE (In this case, "≤" means "<".)

* In the case of WISE ≤ WISE, "≤" means "=".

Try various combinations. However, remember that the character string can only be a maximum of 7 characters.



● Character string size concept

In numbers, the sequence runs from 0 to 9, etc. In the case of letters of the alphabet from A to Z we attach code numbers in ascending order.
Therefore, when we talk about comparing the size of the characters, we are actually talking about the size of the code numbers.
Also, when the number of characters is two or more and the first characters in both names are identical, the codes of the second characters are compared.
So if we compare FOX and FAGAN, FOX > FAGAN.
If we compare AB and ABC, AB ≦ ABC. (In this case, "≦" means "<".)
In this manner, when the number of characters which are compared do not coincide, the last character of the shorter character string is given the smallest value which is called null and then the comparison is made.
Also, in the case of capital letters and small letters, remember that the capital letters have a smaller value.
For example, Fox > FOX.

Let's use the program in Section 4.3 to make a program to sort ABC.

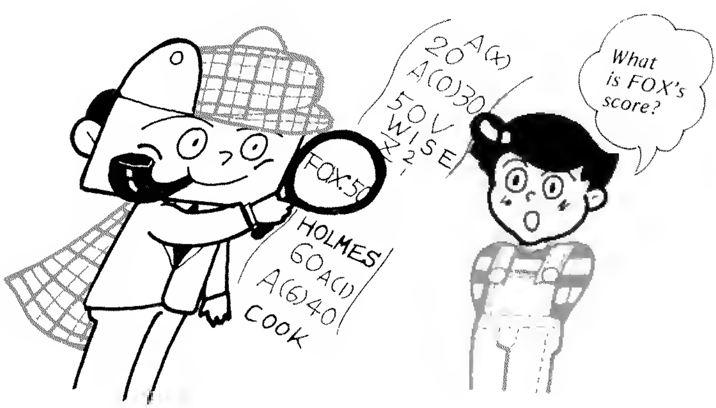
Note

◇ How to use the SET function in a program ◇ (Refer to page 117.)
The SET function can be used in a program.
For example, if it is used in the following manner, the result will be displayed as 0.70711 with five decimal places by rounding off the 6th decimal place.

```
10 SET F5
20 X=SIN(45)
30 PRINT X
40 END
```

4.16 Find the data ! —1

You probably know by now, from the examples given so far, that even though the PC-4 is small, it can store various numerical values and character strings.
Also, another big feature of the PC-4 is that, even though the power is turned off, the contents remain intact.*
That is, whenever you turn the power switch off and on, the programs remain.
Using this feature, let's make a data search program. Searching for some required data from among the stored data is called "search".
Using the previous example, let's make a program to call out the scores of students which were stored along with the names.



*NOTE: However, the programs will be erased when replacing the batteries.

● Determining variables

The first thing we need to consider is how to determine the variables.
Since the PC-4 is an ultra compact computer, the amount of data that can be stored is limited by its size.
The variables which can be used with the PC-4 are normally 26, ranging from A through Z. Additionally, as discussed in the Note on Page 103, even these may be expanded using the DEFM function to a maximum number of 94.
However, if the number of variables is increased to 27 or more, the program storage area is reduced. This is a shortcoming of the PC-4 but this cannot be avoided due to its compact size. On the other hand, this is outweighed by the fact that the PC-4 is inexpensive, compact and that it has a splendid capability for making and using computer programs.
Well, let's take a look at storing the names and scores of 10 students.
For this purpose, as in Section 4.14, let's use array variables. We will use variables A(0) through A(9) to input the scores and character variables K$(0) through K$(9) to input the names.
At this point, you should recall the explanation of array variables which was made on page 99.
Array variables A(0) through A(9) are the same as variables A through J, aren't they? Also, character variables K$(0) through K$(9) are the same as using variables K through T. This means we are using a total of 20 variables.
Therefore, if we do not expand the variables, the number of remaining variables will be 6.

Let's try to use 26 or less variables from A through Z without using the DEFM function.

As a result, the variables which can be used in the program for such things as FOR ~ NEXT number only 6 ranging from U through Z. Please remember this.

● **Program for inputting the data**

This program can be considered to be similar to a sort program, but we will make a few changes.

The concept is as follows.

(1) Count U. Make it 0 initially.
(2) Put data into A(U) and K$(U).
(3) Make U = U+1 and if U=10, terminate. (GOTO (6)).
(4) Ask if data input is complete, using (Y/N). If Y, then terminate. (GOTO (6)).
(5) Return to (2) and repeat.
(6) Terminate (RETURN).

List20 a Input program (INPUT PRO)

```
200 PRINT "INPUT PRO"
210 U=0
220 PRINT U;":NAME";:INPUT K$(U)
230 INPUT"DATA=", A(U)
240 U=U+1 : IF U=10 THEN 270
250 INPUT "END Y/N ", Z$
260 IF Z$ <> "Y" THEN 220
270 RETURN  — Return to main routine
```

Up to 10 inputs can be made.

Since the computer will ask Y/N at each input, inputting may be stopped at less than 10 data.

The "END␣Y/N␣" statement on line 250 means that the computer is asking if input is complete (YES) or not complete (NO).

If Y is input, the program will jump to line 270. If N is input, the program will jump to line 220.

● Verification display program

The input data can be displayed for verification. This is also a subroutine.

List20 b Display Program (PRINT PRO)

```
300 PRINT "PRINT PRO"
310 FOR X=0 TO U−1
320 PRINT X ; ":" : K$(X); "="; A(X)
330 NEXT X
340 RETURN
```

Since it begins with X=0, it will repeat until it reaches U−1

In this program, the number of data previously input determines the number of names and scores which will be displayed.

● Search program

This subroutine compares the character strings for the names and, if they are the same, it displays the name and score.

List20 c Search Program (SEARCH PRO)

```
400 PRINT "SEARCH PRO"
410 V=0
420 INPUT "NAME=", Y$
430 IF Y$ ≠ K$(V) THEN 450
440 PRINT K$(V); ":"; A(V): GOTO 470
450 V=V+1
460 IF V<U THEN 430
470 RETURN
```

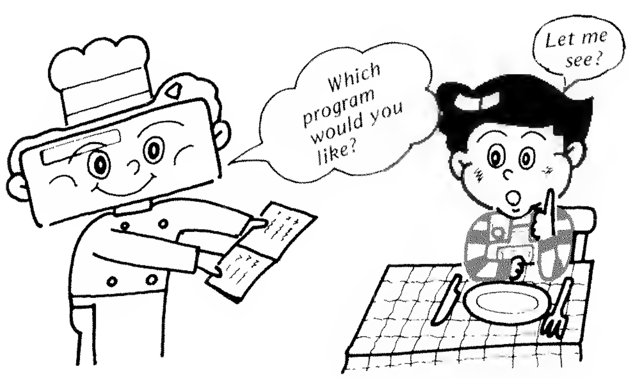
In this program, the names are compared on line 430 and, if they do not coincide, the comparison will be repeated according to U, which is the number of input data. If the name is still not found, the program goes to line 470 and returns to the main routine and terminates.

In this case, you will only know that the program was terminated and will not know why. For this, we add line 465 as below.

```
465 PRINT "NOT FOUND!"
```

This will explain that the program was terminated because the input name could not be located.
So far, we have made three subroutines for data input, display and search. As was explained when discussing the sort program, it is now necessary to make a main program to operate these subroutines.

● **Main routine for data input, display and search**
Let's consider the work flow for this program. The first task is inputting the names and scores. The next task is the display of all input data. The final task is to input the name and display the student's score as required.
However, these three tasks must be performed separately. For example, when inputting and displaying data for confirmation, when displaying all of the data to see what data has been input and when only performing search.



Therefore, it is necessary to be able to select the required task (subroutine).
In order to be able to select one of these, we will assign the following.
"I" for "INPUT PRO" → Data input
"P" for "PRINT PRO" → Display
"S" for "SEARCH PRO" → Search
This kind of program is called a "menu program". The word "menu" here is similar to a restaurant menu.

List20d

Menu program

Program	Comment
`10 INPUT "(I/P/S)", W$`	
`20 IF W$="I" THEN 60`	Selection of data input
`30 IF W$="P"THEN 70`	Selection of data display
`40 IF W$="S" THEN 110`	Selection of search
`50 GOTO 10`	If other than I, P or S is input, returns to beginning
`60 GOSUB 200`	To data input program
`70 GOSUB 300`	To data display program
`80 INPUT "END Y/N ", Z$`	If data input is correct, input Y and terminate.
`90 IF Z$≠"Y" THEN 60`	
`100 END`	
`110 GOSUB 400`	To search program
`120 END`	

This completes the entire program.
Write Lists 20(a) through (d) into the PC-4. The programs are getting longer and more difficult but try your best anyway.
After write-in is complete, let's execute the program.

● **Program execution – Data input and confirmation**

Let's prepare for program execution. If the program was written in program area P0, make sure READY P0 is displayed, then if you press the S R EXE Keys, the following will be displayed.

(I/P/S) ?

This is the menu.
If you press the I Key then press the EXE Key, the following will be displayed.

INPUT PRO

This tells you that this is the beginning of the input program.
Then if the EXE Key is pressed, the following will be displayed.

First, input the data and store them in the PC-4.

0 : NAME = ?

Then, if you input a name and press the EXE Key, the following will be displayed.

DATA = ?

Then, if you input a score and press the EXE Key, the following will be displayed.

END Y/N ?

At this point, if you want to make more inputs, input N(NO) and press the EXE Key. If this is done, the following will be displayed and the cycle can be repeated. 1 : NAME = ?
Since this program has 10 variables each for scores and names respectively – namely, A(0) through A(9) and K$(0) through K$(9) – up to 10 data groups can be input.
Of course, you do not have to input 10 sets of data and can terminate after any number of inputs by inputting Y when the "END Y/N" display is made.

Now, for testing, let's input the following data.
Since we will search for these later, the spelling of the names must be correct.

Number	Name	Score
0	FOX	30
1	WISE	40
2	VALLEY	40
3	MATHIS	70
4	KELLY	40
5	JONES	70
6	HUNT	50
7	HOLMES	60
8	DENT	50
9	COOK	80

Remember names containing up to 7 letters can be input into character variables.

Be sure to spell the names correctly.

After inputting all of the above, the following will be displayed.

PRINT PRO → Display program

Make sure the input data is correct.
Each time you press the EXE Key, the data will be displayed. Upon completion, the following will be displayed.

```
END Y/N ?
```

If a mistake has been made, input N(NO) and the following will be displayed.

```
INPUT PRO
```

This will permit input from the beginning for correction purposes.
If correct, input Y(YES) and the program will terminate.
This completes the data input and confirmation tasks.
Next, we will proceed to the search task.
This is used when you want to know what score FOX made, what score WISE made, etc.

● Program execution – Search task

Press the following keys again and call out the menu.

S RUN (R) EXE

This is a search, so press the S Key and then press the EXE Key. Then, the following will be displayed.

SEARCH PRO

This tells you that this is the beginning of the search program.
At this point, if you press the EXE Key, the following will be displayed.

NAME= ?

This is a request to input the name. In other word, the computer is asking you, "Whose score do you want to know?"
At this point, if you input a name, for example, HOLMES, and press EXE , the following will be displayed as if by magic.

HOLMES :60

Try this by inputting several of the names you input previously.
The PC-4 will compare the currently input name with the previously input names and search for it.
Suppose you input a name which is not contained in the data. For example, input FOY.
Then the following will be displayed.

NOT FOUND!

This tells you that such a name could not be located.
This program concept has many practical uses.
Since the PC-4 is very compact, it can be carried in your pocket. For example, you can input the telephone numbers of your friends.

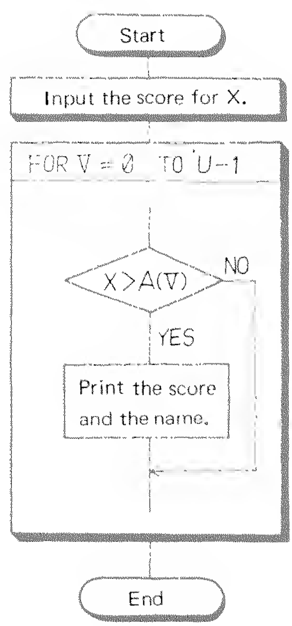
4.17 *Find the data !* —2

The program discussion in the previous section was a bit long. Are you tired? Take a rest before proceeding.

In the previous section, we made a search program to display the names and scores of students when the names were input.
Now let's make a program to display the names and scores of the students whose scores are higher than the input score.

● Program to search for names using scores

The program concept is shown in the flowchart on the right.
At this time, we will make a program which is different from the program in the previous section.



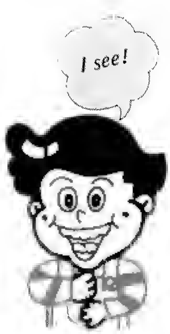
List21

```
500  PRINT"MARK PRO"
510  INPUT"MARK=", X
520  FOR V=0 TO U−1
530  IF X>A(V)THEN 550
540  PRINT K$(V);"=";A(V)
550  NEXT V
560  END
```

The MARK PRO on line 500 means that this is a score program.

On line 520, the FOR ~ NEXT command repeats the comparisons up to the number of data which was input in the previous section. On line 530, the input score is compared with each student's score and if X is greater than A(V), another comparison is made. If X is equal to or less than A(V), a display is made using line 540.
If the IF statement on line 530 is changed as follows, various displays can be made.

Display scores which are equal to or greater than X	IF X>A(V) THEN550
Display scores which are greater than X	IF X≧A(V) THEN550
Display scores which are equal to or less than X	IF X<A(V) THEN550
Display scores which are less than X	IF X≦A(V) THEN550
Display scores which are equal to X	IF X=A(V) THEN550
Display scores which are unequal to X	IF X≠A(V) THEN550



At this time, it should be mentioned that it is very important to insure that the symbols such as $>$, $\geq$, $<$, $\leq$, $\neq$ and $=$ are used correctly. This is because the program flow is opposite to the meaning of these symbols.

● **Program write-in and execution**

Now, select a line 530 format to correspond to your desired operation and write List 21 into the PC-4. However, write it into a program area which is different from the program area used for the program in the previous section.
If you write this program in the same program area as the previous one, List 21 cannot be executed without executing List 20.
After writing, let's execute the program.
For example, suppose List 20 is written in P0 and List

21 is written in P1. Call out the P1 program and execute it.
If we do this, the following will be displayed.

MARK PRO

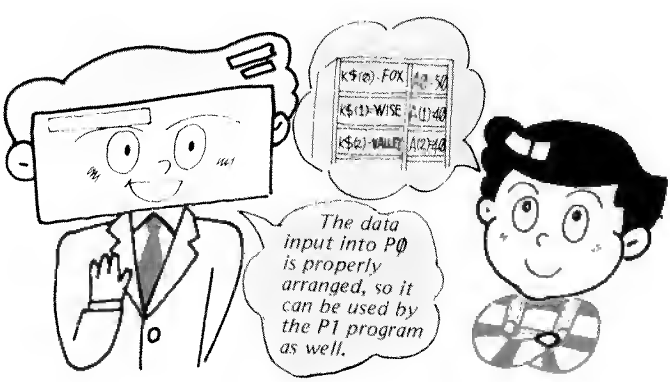
This tells us that this is a program to display the names using the scores.
Next, if [EXE] is pressed, the following will be displayed.

MARK = ?

This is a display which asks, "What is the score?" Suppose 50 is input. If line 53Ø from List 21 is written into the program, names and scores of those students whose scores are 50 or greater will be displayed one after another as follows.

```
50 [EXE] FOX=5Ø
   [EXE] MATHIS =7Ø
   [EXE] JONES =7Ø
   [EXE] HUNT=5Ø
   [EXE] HOLMES =6Ø
   [EXE] DENT  =5Ø
   [EXE] COOK =8Ø
```

In this manner, the data which was input in PØ can also be used for the P1 program.



Call out List 21 using the menu

The previous program was written in the P1 program area separately from List 20. Let's use the List 20 program menu to make it possible to call out List 21. For this purpose, let's make List 21 into a subroutine as follows.

List22

```
500 PRINT "MARK PRO"
510 INPUT "MARK=", X
520 FOR V=0 TO U-1
530 IF X > A(V)THEN 550
540 PRINT K$(V): "=": A(V)
550 NEXT V
560 RETURN
```

The only difference between Lists 21 and 22 is that a RETURN statement has been added using line 560. This makes List 21 a subroutine.
Then, in order to call out this subroutine, the main routine of List 20 (d) must be changed as follows.

List23

```
10 INPUT "(I/P/S/M)", W$
20 IF W$="I" THEN 60
30 IF W$="P" THEN 70
40 IF W$="S" THEN 110
45 IF W$="M" THEN 130
50 GOTO 10
60 GOSUB 200
70 GOSUB 300
```

Program to perform name search using scores.

```
 80  INPUT "END␣Y/N␣", Z$
 90  IF Z$ <> "Y" THEN 60
100  END
110  GOSUB 400
120  END
130  GOSUB #1          To the P1 program area
140  END
```

Only the statements written in blue have been added to this program. "M" was used to represent MARK (Score).
If the statement on line 45 is applicable, the program will jump to line 130 and execute GOSUB #1.
The GOSUB #1 on line 130 means "jump to the P1 program area". If list 22 is written in the P2 program area, make this #2 instead of #1.
This is used similar to GOTO #1 and permits jumping to the beginning of another program area.

If we compose a single program which combines data input, display, search using names, search using scores and this main routine, we will have made a very long program.
However, making the program up a segment at a time is easier than trying to compose the whole program at one time.
Then later, the segments can simply be linked together. This is a useful technique for making programs.

4.18 Roulette is also useful for mathematics !

The computer patiently performs simple, repetitious tasks that humans find tiring. It has already been explained several times that this is one of the greatest features of a computer.
We are fortunate to have the PC-4 to perform these tasks.

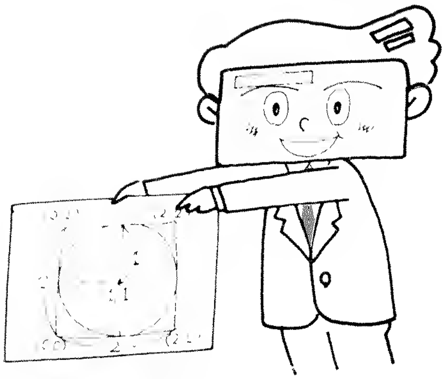
● **"The Monte Carlo Method"**

Now let's make a program to obtain π using the PC-4. Since the PC-4 has a π Key, the value can be obtained immediately by pressing the following Keys.

S π EXE

The display will read 3.141592654.
However, think of this as a kind of game in which random numbers (RAN#) play the main role.
Let's call the game "The Monte Carlo Method". This is a game for obtaining π using probability calculation.

First, let's consider a square whose sides measure 2, such as the one shown in the figure on the next page. (This could be two of anything, for example, 2 meters, 2 centimeters, etc.)
Then, draw a circle inside the square which touches all four sides. In this manner, the diameter of the circle will be 2 and the radius will be 1.



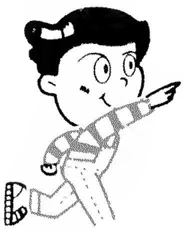
Next, let's throw small rocks randomly inside the area of the square.
If we do this, the rocks will hit points inside the square. However, the ratio of the number of rocks which fall inside the circle to the total number of rocks which were thrown will be the same probability as the ratio of the area of the circle to the area of the square.
In other words, the areas of the square and the circle are as follows.
Area of the square $S_1 = 2 \times 2 = 4$
Area of the circle $S_2 = \pi \times 1^2 = \pi$
The ratio is as follows.

$$S_2 : S_1 = \pi : 4$$

Do you follow up to this point? The person throwing the stones should not be someone with good control. A person with no control should throw a lot of rocks into the square without caring where they fall.
Let's assume that this kind of person throws 40,000 rocks. If so, the formula to obtain the number of rocks which fall inside the circle is composed as follows.

$$40{,}000 \times (\pi/4) \fallingdotseq 31{,}400$$

So we can predict that approximately 31,400 rocks will fall inside the circle.



Taking the opposite approach, if you throw N number of rocks and M number of rocks fall inside the circle, the formula will be as follows.

$$N \times (\pi/4) = M \rightarrow \pi = 4 \times M/N$$

However, 4 x M/N won't always be the value of π (3.141592654). Since N number of rocks fall at random, it is not strange that the actual result will not always coincide with the predicted value.
Therefore, we shall make use of a value which may not be π but is close to π. (This is called the "approximate value".) We will call this value "P". We can make a formula as follows.

$$P = \frac{4M}{N} \quad \cdots\cdots\cdots\cdots (1)$$

Now we will proceed to the main problem.
What we will attempt to do is to use the PC-4 to count how many of the N number of rocks thrown at random will fall inside the circle. Then we will substitute that number for M and obtain the value of P using formula (1).
It may seem like useless repetition but obtaining this value of P is an experiment to obtain the value of π. (This is called "simulation".)
By the way, throwing rocks at random is one of the special capabilities of the computer. Simply generate random numbers as mentioned in a previous example. Random numbers are similar to gambling where the outcome cannot be predicted.
We call simulation using random numbers "The Monte Carlo Method". As you may know, Monte Carlo is a place that is noted for gambling.

The algorithm which obtains the value for P (≒ π)

First, we have to count the number of rocks that fall inside the circle. For this purpose, we have to determine whether or not they fell inside the circle. If you understand the concept, the program to obtain P can be made right away.

To compose a program procedure or calculation method is called an "algorithm".

Now, take a look at the figure below.

Taking the lower left corner of the square as the origin point in an X, Y graph, if we think of the horizontal line as the X axis and the vertical line as the Y axis, each point inside of the square can be expressed using a set of X, Y coordinates.

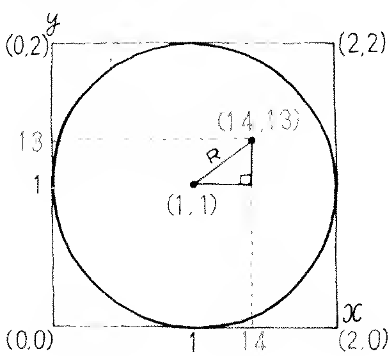
For example, the four corners of the square can be expressed as (0,0), (2,0), (2,2) and (0,2) and the center of the circle can be expressed as (1,1).

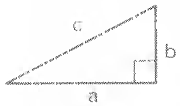
So, let's generate two random numbers for X and Y.

However, these random numbers must be numbers which fall inside the square. Therefore, the following conditions are necessary.

$$0 < X < 2 \qquad 0 < Y < 2$$

For example, when X = 1.4 and Y = 1.3, if the length of a line (R) drawn from the center of the circle (1,1) to the point formed by these coordinates (1.4, 1.3) is shorter than 1 (the radius of the circle), the point will be inside the circle. If R is greater than 1, the point will be outside the circle.





$c = \sqrt{a^2 + b^2}$

This is the famous Pythagorean theorem!

This length R can be obtained using the famous Pythagorean theorem.

The distance of the coordinates from the center of the circle along the X axis is 1.4 – 1 = 0.4 and the distance along the Y axis is 1.3 – 1 = 0.3. Therefore the formula will be . . .

$$R = \sqrt{(1.4 - 1)^2 + (1.3 - 1)^2} = 0.5$$

In other words, since we know that R is less than 1, we can determine that the point is inside the circle.

If this is expressed by a formula using the x, y coordinates, the formula will be as follows.

$$R = \sqrt{(x - 1)^2 + (y - 1)^2}$$

Furthermore, if $x - 1 = X$ . . . . (2)
and $y - 1 = Y$ . . . . (3)
then $R = \sqrt{X^2 + Y^2}$ . . . . (4)

Then, the condition for falling inside the circle will be as follows.

$R < 1$ . . . . (5)

Now we know the algorithm which determines whether a point is inside or outside the circle. The other approach is not so difficult if we use the programming technique which was mentioned previously.

We will now show the program list to obtain P and then take a more detailed look.

List24

```
                        10  GOTO 200
                       100  M=0
                       110  FOR C=1 TO N
From formula (2) ····· 120  X=2*RAN#-1          ··· Random number is doubled
                                                     to obtain a random number
From formula (3) ····· 130  Y=2*RAN#-1               between 0 and 2.
From formula (4) ····· 140  R=SQR(X*X+Y*Y) ···· Distance from the center of
                                                 the circle.
From formula (5) ····· 150  IF R>1 THEN 170 ···· Determines if the point is
                                                 inside or outside the circle.
                       160  M=M+1 ····· Counts if R ≤ 1.
                       170  NEXT C
From formula (1) ····· 180  P=4*M N
                       190  RETURN
                       200  PRINT"SIMULATION PI"
                       210  INPUT "N=", N
                       220  GOSUB 100
                       230  SET F4
                       240  PRINT "PI=" ; P
                       250  GOTO 200
```

Line numbers 100 through 190 are the subroutine of "The Monte Carlo Method" program.
Line 10 and lines 200 through 250 are the main routine to operate the subroutine and display.
In the subroutine, formulas (1) through (5) which were obtained on the previous page are used just as they are.
Also, line 230 is a SET to display four decimal positions. Write this list into the PC-4.

Program execution

Well, let's execute the program.
First, input RUN and press EXE . The following will be displayed.

```
SIMULATION Pi
```

Then, press EXE again and the following will be displayed.

```
N= ?
```

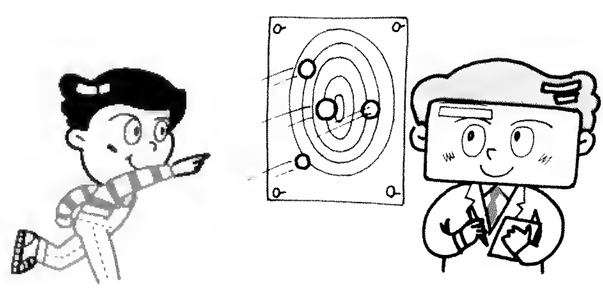
The more random generations used, the greater the probability that the value of Pi will be close to π.
If 1000 is input for N, after two or three minutes, the display will show.

```
Pi =3.1200
```

Inputting 1000 for N each of five times, the following results were obtained.

If you have time, try this by putting 10,000 into N. This may take about 20 minutes.

Number of times	Pi
1	3.1200
2	3.0600
3	3.1200
4	3.1400
5	3.1660



4.19 Let's use the PC-4 to play the "paper-scissors-rock" game.

We are almost to the end of the BASIC Beginner's Manual. Since you have now learned the main commands which are used in PC-4 BASIC, let's conclude by playing a game.

Technique using random numbers

Computer games are all based on random numbers. (This was explained several times before.)
Suppose we play the "paper-scissors-rock" game with the PC-4. Since this game is composed of three words (or symbols), "paper", "scissors" and "rock", let's treat them as numerical values, namely, "0", "1" and "2" respectively.
Then the computer's task is to generate these three numbers in random sequence.
The generation of 0, 1 and 2 is performed as follows.

Makes the contents of the parentheses into the integer.

```
X = INT (3*RAN#)
```

Generates a random number between 0 and 1

Why do we need to multiply by 3? This is a technique when using random numbers.
The relationship for multiplying the random number by 3 and obtaining the integer part is explained as follows.

RAN#	Times 3	INT
0.000000000	0.000000000	0
∫	∫	
0.333333333	0.999999999	0
0.333333334	1.000000002	1
∫	∫	
0.666666666	1.999999999	1
0.666666667	2.000000001	2
∫	∫	
0.999999999	2.999999997	2

In this manner, values of 0, 1 and 2 are generated with a probability of 1/3.
Also, in the "electronic dice" game on page 113, we output numerical values from 1 through 6 using the following.

```
X=INT(10*RAN#)+1
IF X>6 THEN 200
```

Statement to disregard numbers above 6.

However, numerical values from 1 through 6 can be obtained using only one line as follows.

```
X=INT(6*RAN#)+1
```

Why do we have to multiply by 6 and add 1? Please study this and answer the question by yourself.

- **The algorithm for the "paper-scissors-rock" game**

The objective is what kind of "paper-scissors-rock" game should be made.
At this point, we will simply display the result each time the game is played.
In order to do this, we must determine the rules.

The algorithm (program concept) is shown below.

(1) N = 0: M = 0
N: Number of times the PC-4 wins.
M: Number of times you win.

(2) Input your selection into X.
0 = paper
1 = scissors
2 = rock

(3) Input the PC-4 selection into Y.

(4) Compare X and Y.
When X = Y, tie
When X = 0 and Y = 1, you lose.
Y = 2, you win.
When X = 1 and Y = 0, you win.
Y = 2, you lose.
When X = 2 and Y = 0, you lose.
Y = 1, you win.
When you win, M = M + 1
When the PC-4 wins, N = N + 1

Y \ X	0	1	2
0	TIE	WIN	LOSE
1	LOSE	TIE	WIN
2	WIN	LOSE	TIE

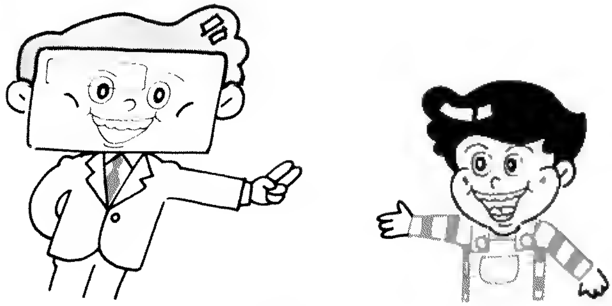
(5) Return to (2).

If you try to make the game more complicated, you can make it more interesting. However, we will make a program with the basic portion of the algorithm only. Later, you can make various modifications to this program.

Program for the "paper-scissors-rock" game

List25

```
10  N=0:M=0
100 PRINT "PAPER-SCISSORS-ROCK"
```



```
110  PRINT "YOUR  SELECTION"
120  INPUT "0 : PAPER  , 1 : SCISSORS
     2 : ROCK", X
130  IF X<0 THEN 110        No result, return to the beginning
140  IF X>2 THEN 110        Void. Return to starting point.
150  Y=INT(3*RAN#)          Determines the PC-4's
                            selection
200  IF X=Y  THEN 300
210  IF X=0  THEN 400
220  IF X=1  THEN 500
230  IF X=2  THEN 600
300  PRINT "TIE" : GOTO 110
400  IF Y=1  THEN 750       LOSE
410  IF Y=2  THEN 700       WIN
500  IF Y=0  THEN 700       WIN
510  IF Y=2  THEN 750       LOSE
600  IF Y=0  THEN 750       LOSE
610  IF Y=1  THEN 700       WIN
700  PRINT "YOU  WIN" : M=M+1
710  GOTO 110
750  PRINT "YOU  LOSE" : N=N+1
760  GOTO 110
```

In this program, since the number of wins and losses are counted on lines 7ØØ and 75Ø, the total number of trials, for example, 100 times, can be computed.
Then, the number of times you won or lost can be displayed.
We will leave it up to you as to how to make this kind of program.

This completes the beginner's guide to BASIC.
Those who have even roughly mastered some of the uses of BASIC up to this point can become more skilled by constant practice.
We have stated previously that with computers, doing is better than just reading. To become more skilled at programming, you should make many more programs by yourself. Then, as shown in Chapter 5, you should refer often to programs made by others. By the way, we might mention again that to be able to use full-fledged BASIC with such a compact computer as the PC-4 is truly amazing.
As was stated in the beginning, the BASIC language has many dialects, each being slightly different.
Therefore, there is never 100% compatibility between two different systems that use BASIC.
However, if you understand the programming concepts used here, it should be easy to become accustomed to other dialects of BASIC.
In addition, it will become easier to use larger personal computers. So use the PC-4 as a starting point to become skilled at programming in BASIC.
Not all of the functions of the PC-4 were explained in the previous sections. For example, the PC-4 also has the capability of increasing its RAM size by adding a RAM Expansion Pack, and also to receive and send data through a taperecorder interface into magnetic tape.

In addition, the BASIC grammar of the PC-4 has more functions than have been stated here. The majority of these will be explained in the Instruction Manual, so please refer to it.
These should be easy to understand for those who have already read up to this point.

Chapter 5

Program Library

Cassette tape recording plan

<CT-TIME>

Method to fully record on a cassette tape.

Preparation prior to program input.

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 0 [EXE]

```
P0
  10 VAC
  20 PRINT "CT-TIME"
     :SET N
  30 INPUT "TAPE LEN
     GTH",A
  40 I=1
  50 PRINT "NO:";I
  60 INPUT "TIME:M.S
     ",B
  70 C=C+INT B
  80 D=D+FRAC B/.6
  90 IF A≤C+D THEN 2
     00
 100 E=C+D
 110 I=I+1
 120 GOTO 50
 200 PRINT "FULL NO.
     ";I-1
 210 F=INT (A-E)+FRA
     C (A-E)*.6
 220 SET F2
 230 PRINT F:GOTO 10
```

Total 182 steps

Memory contents	
A	Tape length (minutes)
B	Length of 1 selection (minutes, seconds)
C	Accumulation of minutes portion
D	Accumulation to convert seconds into minutes
E	Accumulated time for each selection (minutes)
F	Remaining time (minutes, seconds)
I	Selection number counter

Explanation This is a program which obtains the remaining time and number of selections which can be recorded on one side of a cassette tape. First, input the length of one side of the tape in minutes. Next, input the minutes and seconds for each selection. At this time, AA(minutes)and BB(seconds) are input as AA [•] BB.
If the total selection time exceeds the length of one side of the tape, the program terminates and displays the total number of selections which can be recorded along with the remaining time.

Example 46-minute tape (23 minutes on one side)

A.1	3 : 36	B.1	2 : 30
2	4 : 56	2	3 : 15
3	3 : 30	3	5 : 10
4	2 : 59	4	6 : 20
5	4 : 00	5	1 : 51

The problem is how to effectively edit one tape.

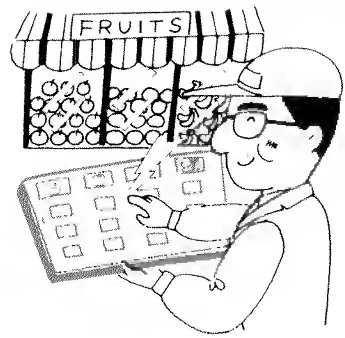
Operation

Step	Key operation	Display
	[MODE] [0]	
1	[S] [PU/0]	CT−TIME
2	[EXE]	TAPE LENGTH?
3	23 [EXE]	NO:1
4	[EXE]	TIME:M.S ?
5	3.36 [EXE]	NO:2
	After repeating from step 4, at No. 7, FULL NO. 6 will be displayed. Press [EXE] and 1.29 will be displayed. This means that the selections from A.1 to B.1 can be input and the time remaining will be 1 minute, 29 seconds.	

Vertical and horizontal total

(CROSS TOTAL)

This program is useful for a variety of fields. We no longer need totals tables.



Preparation prior to program input

MODE 1 CLEARA EXE

S DEFM 40 EXE

```
P0
  1 VAC
  2 PRINT "CROSS TO
    TAL":INPUT "X-I
    TEM NO. ",A:IF
    A>58 THEN 2
  3 B=B+1
  4 FOR D=1 TO A
  5 PRINT "Y";B;"-X
    ";D;:INPUT E
  6 C=C+E:F=F+E:G(D
    )=G(D)+E
  7 NEXT D
  8 PRINT "Y";B;"="
    ;C:C=0:GOTO 3

P1
  1 FOR D=1 TO A
  2 G=RND(G(D)/F,-3
    )*100
  3 PRINT "X(";D;")
    =",G(D);"(";G;"
    %)"
  4 NEXT D
  5 PRINT "GRAND T.
    =",F:GOTO 1
```

Total 541 steps

Memory contents	
A	Number of X (horizontal) items
B	Y (line) number counter
C	Horizontal total
D	Horizontal total control variable
E	Input data
F	Grand total
G	Composite ratio

Explanation

First, determine how many horizontal axes (X) are required. If you input the numbers which correspond to the vertical items (Y), the Y line subtotals will be displayed using the designated X numbers. After completing the required number of vertical axis lines, each time you press the [EXE] Key, the X axis subtotals and percentages will be displayed one after another. Therefore, the X axis represents the items for which the composite ratio is required.

PØ : Total of respective horizontal items: memory G(1) to G(59). Up to 59 items are permitted for X to display the horizontal item subtotals

P1 : Respective totals display and composite ratio display

Example

	X_1	X_2	X_3	X_4	X_5	Total
Y_1	23	26	2	50	32	133
Y_2	19	46	11	19	10	105
Y_3	79	54	22	30	86	271
Y_4	35	11	15	12	5	78
Y_5	19	11	39	20	21	110
Total	175	148	89	131	154	697
Composite ratio (%)	25	21	13	19	22	--

Operation

Data input

Step	Key operation	Display
	[MODE] [Ø]	
1	[S] [PØ]	CROSS TOTAL
2	[EXE]	X-ITEM NO. ?
3	(Ex.) 5 [EXE]	Y1-X1 ?
4	(Ex.) 23 [EXE]	Y1-X2 ?
5	26 [EXE]	Y1-X3 ?

Step	Key operation	Display
6	After inputting Y1-X5 EXE	Y1=133 Y2−X1 ?
	Keep repeating through Y5-X5	

To see the vertical axis total and its ratio after inputting all data.

Vertical axis total and composite ratio

Step	Key operation	Display
1	S P1 (1)	X(1)=
2	EXE	175(25%)
3	EXE	X(2)=
	Hereafter, keep repeating steps 2 and 3 and finally, "GRAND T. =" will be displayed.	

Scheduler

<SCHEDULER>

Let's perform schedule management using a machine. It's kind of like science fiction, isn't it?

Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 21 [EXE]

```
P0  1 GOSUB #4:C=5
    2 FOR B=0 TO 21
    3 IF INT D(B)=A;PRINT D(B),Z$(B):C=1
    4 NEXT B:GOTO C
    5 PRINT "NO SCH.":GOTO 1
P1  1 GOSUB #5
    2 FOR B=0 TO 22
    3 IF B>21;PRINT "MEMO OVF":GOTO 1
    4 IF D(B)=0;D(B)=A:GOTO 6
    5 NEXT B
    6 INPUT "SCH.: ",Z$(B):GOTO 1
P2  1 PRINT "SCH.:",Z$(B),"DEL. OR NOT"
    2 INPUT "D OR N ",C$:IF C$≠"D";GOTO #1
    3 C=B
    4 FOR B=C+1 TO 21
    5 IF D(B)=0 THEN 7
    6 NEXT B
    7 IF B=1 THEN 9
    8 D(C)=D(B-1):Z$(C)=Z$(B-1)
    9 D(B-1)=0:GOTO #0
P3  1 PRINT "SCHEDULER":SET F4
    2 VAC
    3 GOTO #1
P4  1 INPUT "DATE ",A:RETURN
P5  1 GOSUB #4:INPUT "TIME ",B:A=A+B/10↑4:RETURN
```

Total 544 steps

Memory contents	
A	DATE input/+TIME
B	TIME input
C	Readout of DEL. Or NOT and switching pointer for SCH.

Memory contents					
D ~ Y	D$ (0) ~ D$(21)	Beginning of DATE table. DATE table uses 22 variables from D$ (0) through D$ (21) = Y$. 1 variable per item.	Z	Z$ (0) ~ Z$(21)	Beginning of SCHEDULE table. SCHEDULE table uses 22 variables from Z$ (0) = Z$ through Z$ (21). 1 variable per item.

Explanation By inputting month and day, you can list a maximum of 22 schedule items. The PC-4 will display DATE? TIME? SCH.?. Make the desired inputs. When referring to the schedule, if you input the desired month and day, the entire schedule for that day will be displayed. You can make additions and deletions freely. PØ through P3 are lined up in file deletion risk sequence.

PØ : Reference. If you input the date, the schedule for that day will be searched and displayed in sequence. If there is no further schedule, the program returns to the beginning. If there is no schedule, "NO SCH." will be displayed.

P1 : Addition. Corresponding to the displayed request, input the date, time and schedule item. For input, line up the characters and press [EXE] .

P2 : Deletion. To delete a particular item, first locate the item using P0 and when that schedule is displayed, switch to P2. The PC-4 will ask you whether you want to delete or not.

P3 : New list. Input at " ? " indicator.

- If listing of more than 22 items is attempted, "MEM OVF" will be displayed.
- If you attempt to input a schedule in excess of 7 characters, ERR 6 will be displayed. In this case, press [AC] and input again.

Operation

Step	Key operation	Display
1	[MODE] [0] [S] P3 [3]	SCHEDULER
2	[EXE]	DATE ?
3	(Ex.) For 15 June, 615 [EXE]	TIME ?
4	(Ex.) For 10 : 00 AM. 1000 [EXE]	SCH. : ?

Step	Key operation	Display
5	(Ex.) MEETING [EXE]	DATE ?
	Continue repeating Steps 3 through 5 and input DATE, TIME and SCH.	

* For 5 January, input 105 and for 1:30 PM, input 1330.
* SCH. can be a maximum of 7 characters.

Reference
P 0

Step	Key operation	Display
1	[S] [P0 0]	DATE ?
2	(Ex.) 615 [EXE]	615.1000
3	(Ex.) [EXE]	MEETING
	If you repeatedly press [EXE], the entire schedule for that day will be displayed.	

Addition
P 1

Step	Key operation	Display
1	[S] [P1 1]	DATE ?
2	(Ex.) 615 [EXE]	TIME ?
3	1830 [EXE]	SCH. : ?
4	(Ex.) TEL [EXE]	DATE ?
	Repeat Steps 2 through 4 the desired number of times.	

Deletion
P 2

Step	Key operation	Display
1	[S] [P0 0]	DATE ?
2	(Ex.) 615 [EXE]	615.1830
3	[EXE]	TEL
4	[S] [P2 2]	SCH. :
5	(Ex.) [EXE]	TEL
6	[EXE]	DEL. OR NOT(Is it O.K. to delete?)
7	[EXE]	D OR N?(Use D or N to answer.)
8	D [EXE]	DATE ? (Return to P0.)

* First, move to P0, display that item, move to P2 and input at " ? " indicator. DEL. means DELETE.

- For an item other than new list, if ([S] [P3 3]) are pressed, all data will be cleared, so be careful.

Telephone list

<T-LIST>

This is a useful telephone list which permits addition and deletion. This could also be made into a price list by changing the names and telephone numbers to items and prices. Other similar lists can be stored.

Preparation prior to program input

MODE 1 CLEARA EXE

S DEFM 21 EXE

```
P0  2 IF A=0;B=14:GOTO 7
    3 GOSUB #9
    4 FOR B=0 TO A-1
    5 IF D$(B)=C$ THEN 7
    6 NEXT B:B=14
    7 $=R$(2*B)+R$(2*B+1)
    8 PRINT $:GOTO 2

P1  3 IF A≥14 THEN 11
    4 GOSUB #9
    5 D$(A)=C$:$="":INPUT "TEL-NO ",$
    6 B=LEN($):R$(2*A+1)=R$(29)
    7 IF B>14 THEN 11
    8 IF B>7;R$(2*A)=MID(1,7):R$(2*A+1)=MID(8):GOTO 10
    9 R$(2*A)=MID(1)
   10 A=A+1:GOTO 3
   11 PRINT "OVF":GOTO 3

P2  1 A=A-1:D$(B)=D$(A)
    2 FOR C=0 TO 1
    3 R$(2*B+C)=R$(2*A+C)
    4 NEXT C:GOTO #0

P3  1 PRINT "T-LIST"
    2 VAC
    3 R$(28)="NO NAME":R$(29)="":GOTO #1

P9  1 INPUT "NAME ",C$:RETURN
```

Total 544 steps

Memory contents		
A		Listed item counter
B		Loop control variable
C		NAME read area
D ⁝	D$(0) ⁝	following 14 variables are for name list ⁝
Q	D$(13)	
R	R$(0)	following 28 variables are for name list
⁝	⁝	⁝

Explanation

If a name is input, the telephone number will be displayed. The total number of steps required for this program is 376. Since the total number of steps in the PC-4 is 544, the remaining 168 steps are assigned for memory. As a result, DEFM (define memory) is 21 and space is available to store the names and telephone numbers of 14 people.
Initial input of names and numbers, referencing, addition and deletion is performed by keying in the name.
By adding the memory (option), it is possible to store information for 50 to 55 people.
To use, depending on the purpose, select programs PØ through P3.

PØ : Telephone list reference (if not listed, "NO NAME" will be displayed.)
P1 : Item addition (Input "NAME" and "TEL-NO" alternately.)
P2 : Deletion. Immediately after referring to the telephone number in the P1 area, that item is deleted and the program returns to the initial point of P1.
P3 : Telephone list file creation. After the variables are cleared, program proceeds to P1.

- Make deletions prior to making changes.
- Perform input after " ? " is displayed.
- When inputting telephone numbers into P1, if you exceed 14 characters, "OVF" (overflow) will be displayed. Also, if you attempt to input more than 14 items, "OVF" will be displayed.

Operation

Listing

Step	Key operation	Display
	MODE Ø	
1	S P3 (3)	T－LIST
2	EXE	NAME ?
3	(Ex.)HOLMES EXE	TEL－NO ?

Step	Key operation	Display
4	(Ex.)06-281-6253 EXE	NAME ?
5	Hereafter, input required items using procedures 3 and 4.	

Reference
P0

Step	Key operation	Display
	MODE 0	
1	S P0 0	NAME ?
2	(Ex.) HOLMES EXE	(Ex.) 06-281-6253

Addition
P1

Step	Key operation	Display
	MODE 0	
1	S P1 1	NAME ?
2	(Ex.)FOX EXE	TEL—NO ?
3	(Ex.)056-714-1855 EXE	056—714—1855 NAME ?

Deletion
P0
P2

Step	Key operation	Display
1	MODE 0	
2	S P0 0	NAME ?
3	(Ex.)FOX EXE	056-714-1855
4	S P2 2	NAME ?
	The name and telephone number in step 3 are deleted and the program returns to P0.	

Train timetable

<TIMETABLE>

This is quite convenient when travelling by train or air. You can avoid bothering with timetables.

Preparation prior to input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 25 [EXE]

```
P0
 1 PRINT "SEARCH"
 2 GOSUB #9:FOR B=
   0 TO 2:IF $=E$(
   B) THEN 5
 3 NEXT B
 4 PRINT "NO TR.NA
   ME":GOTO 2
 5 INPUT "H ",D:FO
   R C=H(B) TO H(B
   +1)-1:IF D≤INT
   L(C);GOSUB 7
 6 NEXT C:GOTO 5
 7 PRINT E$(B);C-H
   (B)+1;" :":SET
   F2:PRINT L(C):S
   ET N:RETURN

P1
 1 PRINT "ENTER":A
   +1:IF A≥3;PRINT
    "TIME TAB OVF"
   :GOTO 1
 2 H(A)=K:GOSUB #9
   :E$(A)=$:A=A+1
 3 IF K≥40;PRINT "
   TIME TAB OVF":G
   OTO 3
 4 PRINT D$(A);K-G
   (A)+1;":":INPUT
    "H.M ",L(K):K=
   K+1:GOTO 3

P2
 1 PRINT "TIMETABL
   E"
 2 VAC
 3 GOTO #1

P9
 1 INPUT "TR.NAME
   ",$:IF LEN($)>7
   ;$=MID(1,7)
 2 RETURN
```

Total 542 steps

Memory contents		
A		Name index
B		Control variable
C		Control variable
D		Time readout buffer
E ∫ G	E$ (0) ∫ E$ (2)	TR. NAME table
H ∫ K	H (0) ∫ H (3)	Time index table Time index
L	L (0)	Time
∫	∫	

Explanation

Using a maximum of 3 categories and the names of 7 characters or less, a total of up to 40 train schedules can be stored. The three categories can be called limited express, express and local, or you might want to use the name of the line.

To call out the contents corresponding to the category requested, if you input the train category or line name, the time will be requested. Then input the time and all time schedules (from that input time) within that category will be displayed in the listed sequence.

P2 : New listing. For a new listing, the PC-4 will ask you TR.NAME, so input train name, train type, line name, etc. as desired. Since the PC-4 will repeatedly ask the time, make inputs as desired and then move to P1. As an example, 11:30 AM is input as 11[•]30.

P1 : Train category addition. Since ENTER 2 is displayed, input another line, etc. and input the time. If more lines or trains are desired to be input, press [S] [1] again.

P0 : Reference. According to the request, if you input the desired TR.NAME, the PC-4 will ask for the time H. If you enter a number, the entire train schedule from that train on will be displayed.

Operation

List
P2, P1

Step	Key operation	Display
	[MODE] [0]	
1	[S] [2]	TIMETABLE
2	[EXE]	ENTER 1
3	[EXE]	TR. NAME ?
4	(Ex.)BULLET[EXE]	BULLET 1 :
5	[EXE]	H.M ?
	(Ex.) 8.36 [EXE]	BULLET 2 :

Step	Key operation	Display
	EXE	H.M ?
	8.48 EXE	BULLET 3 :
9	S P1 1	ENTER 2
10	EXE	TR. NAME ?
11	(Ex.) DART EXE	DART 1 :
12	EXE	H.M ?
13	12.40 EXE	DART 2 :
14	S P1 1	ENTER 3
15	EXE	TR.NAME ?
16	(Ex.) EXP. EXE	EXP. 1 :
17	EXE	H.M ?
18	(Ex.) 13.00 EXE	EXP.. 2 :
19	EXE	H.M ?
20	14.05 EXE	EXP. 3 :

Example to change to another limited express train on the Blue Line

Example to change to an express train on the Blue Line

EXPRESS is abbreviated EXP.

Addition P1

Train schedule additions cannot be made. Addition of train names can be performed in P1. ENTER 3 input exchange can be made by pressing the following keys.

MODE 0 AC A=2 EXE S P1 1

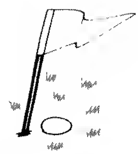
Reference P0

Step	Key operation	Display
	MODE 0	
1	S P0 0	SEARCH
2	EXE	TR.NAME ?
3	(Ex.) BULLET EXE	H ? (After what time?)
4	8.00 EXE	BULLET 1 :
5	EXE	8.36
	Hereafter, continue to press EXE and the listed times will be displayed one after another.	

Golf game

<GOLF GAME>

Golf with a mechanical feeling! The flight path of the ball is represented by revolving numbers.



Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 0 [EXE]

P0
```
10 G=0:N=50
20 PRINT "GOLF GAME";:GOSUB #9:PRINT "BEST-SCORE";N;:GOSUB #9
30 Q(1)=4:Q(2)=5:Q(3)=4:Q(4)=3:Q(5)=4:Q(6)=5:Q(7)=4:Q(8)=3
40 Q(9)=4
50 FOR H=1 TO 9
60 L=Q(H)*20-20+INT (RAN#*10)
70 PRINT H;":PAR";Q(H);L;"M";:GOSUB #9:GOSUB #9:GOSUB #9
80 GOSUB #1:G=G+C:PRINT "(";C-Q(H);")";:GOSUB #9:GOSUB #9
90 NEXT H:IF G<N;N=G:PRINT "** BEST=";G;"**":G=0:GOTO 20
100 PRINT "SCORE=";G
110 G=0:GOTO 20
```

P1
```
10 C=0
20 PRINT L;" PUSH KEY";
30 IF KEY≠"" THEN 30
40 IF KEY="" THEN 40
45 GOSUB #9
50 PRINT L;" SHOT!";:C=C+1:FOR I=1 TO 1E99:IF KEY="" THEN 100
60 NEXT I
100 J=SGN L
110 FOR B=1 TO I:L=L-J:PRINT :PRINT L;:NEXT B
120 IF L=0;RETURN
130 GOTO 30
```

P9
```
10 FOR I=1 TO 100:NEXT I
20 PRINT
30 RETURN
```

Total 433 steps

Memory Contents		
C		Par
G		Total par
H		Hole number
I		Counter (distance of the ball)
J		L mark
L		Distance to the hole
N		Best score
R ~ Z	Q(1) ~ Q(9)	Par for each hole

Explanation This is the same method as actual golf. There are 9 holes. The par for each hole and distance to the hole are displayed. Try to get the ball in the hole with as few strokes as possible.
For each hole, the amount over or under par will be displayed. Also, after completing 9 holes, the total number of strokes will be displayed. The lower the number of strokes, the better.
When hitting the ball, the distance is determined by the length of time that a numerical key is pressed. If you keep pressing for a long time, the ball will overshoot the hole. In this case, the distance to the hole will be shown as a minus.

Operation

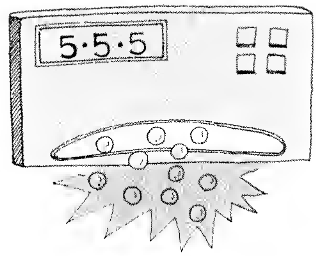
Keep pressing any numerical key.

Step	Key operation	Display
	[MODE] [0]	
1	[S] [PJ 0]	GOLF GAME BEST-SCORE 50 1 : PAR 4 69 M 69 PUSH KEY (1st hole, par 4, distance 69 m)
2	(Ex.) 3	−28
	3	− 2
	3	0 (−1) (Display for the next hole)
	Continue repeating until 9 holes are completed.	
		SCORE 48 If the score is better than the previous best score, the display will be as follows. * *BEST=29* *

Slot game

<SLOT GAME>

The speed with which the 3 numbers change gets slower and slower. This is the part that requires programming technique.



Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [V] (DEFM) 0 [EXE]

```
P0
  5 G=0
 10 PRINT "SLOT GAME"
 20 INPUT "AMOUNT OF BET",A
 30 B=1
 40 FOR C=1 TO 12
 50 X=INT (RAN#*10)
 60 Y=INT (RAN#*10)
 70 Z=INT (RAN#*10)
 80 PRINT CSR 1;X;":";Y;":";Z;
 90 FOR D=1 TO C↑2
100 NEXT D
110 NEXT C
120 IF X=Y;B=B*4
130 IF Y=Z;B=B*4
140 IF X=Z;B=B*4
150 IF B=1;B=-1
160 G=G+B*A:STOP
170 PRINT CSR 1;B*A;"(";G;")"
180 GOTO 20
```

Total 225 steps

Memory contents	
A	Amount of bet
G	Total amount
X	Random number
Y	
Z	

Explanation

Just like a slot machine, 3 numbers can be seen as if they were rotating. The rotation gets slower and slower and finally stops. This is the essence of the game. This is realized by using a double loop of a FOR ~ NEXT statement.

Rules
- 2 like numbers: 4 to 1
- 3 like numbers: 64 to 1

Operation

Step	Key operation	Display
	MODE 0	
1	S P0	SLOT GAME
2	EXE	AMOUNT OF BET ?
3	(Ex.) 100 EXE The amount of the bet is up to you.	Numbers change 12 times then stop.
4	(Ex.) EXE	−100 (−100) Amount in ( ) indicates amount remaining.
5	Repeat from step 2.	

Standard deviation and statistics

<STATI-0>

This covers deviation value calculation and various related calculation data. This has many uses for schools and for quality control.

Preparation prior to program input

[MODE] [1] CLEAR A [EXE]

[S] [DEFM V] 0 [EXE]

```
P0
10 PRINT "STATI-0":IF L$≠"" THEN 30
11 PRINT "COMMENT ?":INPUT "KEY IN Y OR N ",L$
12 IF L$="N" THEN 30
20 PRINT "X:DATA"
21 PRINT "Q:SUM OF SQ-X"
22 PRINT "S:SUM OF X"
23 PRINT "N:NO. OF DATA"
24 PRINT "A:AVERAGE"
25 PRINT "D:SD(N)"
26 PRINT "E:SD(N-1)"
30 INPUT "KEY IN X ",X:N=N+1:S=S+X:Q=Q+X*X:GOTO 30

P1
10 PRINT "Q=";Q:GOTO 10

P2
10 PRINT "S=";S:GOTO 10

P3
10 PRINT "N=";N:GOTO 10

P4
10 A=S/N
20 PRINT "A=";A:GOTO 20

P5
10 D=SQR ((Q-S*S/N)/N)
20 PRINT "SD(N):D=";D:GOTO 20

P6
10 E=SQR ((Q-S*S/N)/(N-1))
20 PRINT "SD(N-1):E=";E:GOTO 20

P7
10 INPUT "H= ",H:Y=(H-50)*E/10+A:PRINT "X=";Y:GOTO 10

P8
10 A=S/N:E=SQR ((Q-S*S/N)/(N-1))
20 INPUT "KEY IN X ",X:H=10*(X-A)/E+50:PRINT "D.VALUE =";H
30 PRINT H:GOTO 20

P9
10 Q=0:S=0:N=0:L$="":GOTO #0
```

Total 512 steps

Memory Contents	
A	$\bar{x}$
D	SD (n)
E	SD (n-1)
H	Deviation value
N	n: Number of data
Q	Σx^2
S	Σx
X	Input data x
Y	x obtained by H

Explanation Concerning deviation value, it has been explained in Chapter 4. In a given group, it is known that data is usually formed as a normal distribution. In this, the concept of a deviation value compensates for ungrouped data. The standard deviation is widely used as a measure in statistical work.

This obtains the necessary calculated value in order to find the characteristics of the group. In addition, if the deviation value is given, you can find the equivalent value. In order to extract the various data, many windows are prepared from P1 through P8.

- P9: Data input. Clears previous data. Since it repeatedly asks you KEY IN X?, after all data is input, specify the required program area.
- PØ: Data input. Since previous data is not cleared, continuously input data. Data is X.
- P1: ΣX^2. Sum of square of X
- P2: ΣX. Sum of X
- P3: N. Number of data elements
- P4: $\overline{X}$ or ΣX / N. Average
- P5: D. Standard deviation σ_N (population)

$$\sigma_N = \sqrt{\frac{\Sigma X^2 - (\Sigma X)^2 / N}{N}}$$

- P6: E. Standard deviation σ_{N-1} (sample)
- P7: H → X. Obtain X by inputting deviation value.
- P8: X → H. Obtain deviation value by inputting X.

Example Obtain ΣX^2, ΣX, N, $\overline{X}$, σ_N and σ_{N-1} for the following data.

25	46	53	64	70	80	92	100
59	67	75	78	85			

Operation

Step	Key operation	Display
1	MODE 0	
2	S 9	STATI-0
3	EXE	COMMENT ?
4	EXE	KEY IN Y OR N ?
5	Y EXE	X : DATA (Individual data)
	If Y (YES) is input, explanation of the symbol will be displayed by pressing EXE. Q : SUM OF SQ-X ($Q=\Sigma X^2$). S : SUM OF X. N : NO. OF DATA. A : AVERAGE. D : SD(N) E : SD(N 1) If N (NO) is input, the above explanations will be skipped by pressing EXE.	
6	EXE	KEY IN X ?
7	25 EXE	KEY IN X ?
	Hereafter, required data is input repeatedly.	

Data input P9

If you forget the meaning of the symbol, make COMMENT YES.

P1 ~ P8

Step	Key operation	Display
8	S 1	Q=66314
9	S 2	S=894
10	S 3	N=13
11	S 4	A=68.76923077
12	S 5	SD(N) : D=19.28392653
13	S 6	SD(N−1) : E= 20.0713471
14	S 7	H= ?
15	(Ex.) 53 EXE	X= 74.7906349
16	S 8	KEY IN X ?
17	85 EXE	D. VALUE = 58.08653707

What is the value for the deviation value, 53?

What is the deviation value for 85?

Regression analysis

〈REGRESSION ANALYSIS〉

Linear, exponential, logarithmic and power regression analysis are all possible.

Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 0 [EXE]

```
P0
  1 S=T/N:B=(V-S*U)
    /(W-S*T):A=(U-B
    *T)/N:C=A:GOSUB
     P+1:GOTO 6
  2 RETURN
  3 GOTO 5
  4 RETURN
  5 A=EXP 1↑C:RETUR
    N
  6 Q=U*U/N:R=(C*U+
    B*V-Q)/(Z-Q):PR
    INT "a=";A,"b="
    ;B,"r↑2=";R
  7 INPUT "x= ",F:I
    F F≤0 THEN 7
  8 GOSUB P*10:PRIN
    T "EXPECTED y =
    ",G:GOTO 7
 10 G=A+B*F:RETURN
 20 E=EXP 1
 21 G=E↑(B*F):G=A*G
    :RETURN
 30 G=A+B*LN F:RETU
    RN
 40 G=A*F↑B:RETURN

P1
  1 VAC
  2 $="LINEAR":P=1:
    GOTO #5

P2
  1 VAC
  2 $="EXPONENTIAL"
    :P=2:GOTO #5

P3
  1 VAC
  2 $="LOGARITHMIC"
    :P=3:GOTO #5

P4
  1 VAC
  2 $="POWER":P=4:G
    OTO #5

P5 3 PRINT "REGRESSI
     ON ANALYSIS";-P
     ;":",$
   4 PRINT "ROUND ";
     N+1:N=N+1:INPUT
      "x=",X,"y=",Y:
     GOSUB P+4:GOTO
     9
   5 RETURN
   6 Y=LN Y:RETURN
   7 X=LN X:RETURN
   8 X=LN X:GOTO 6
   9 T=T+X:U=U+Y:V=V
     +X*Y:W=W+X*X:Z=
     Z+Y*Y:GOTO 4
```

Total 500 steps

Memory Contents	
A	a
B	b
C	c
F	EXPECTED x
G	EXPECTED y
N	n
P	Indirect GOTO pointer
Q	$(\Sigma Y)^2/n$
R	r^2
S	$\bar{x}$
T	ΣX
U	ΣY
V	ΣXY
W	ΣX^2
X	X
Y	Y
Z	ΣY^2

Explanation

This is a method for predicting a statistical data distribution by determining which formula is the closest fit.
Even for data which seems at a glance to be completely random, if analyzed, it can be expressed as an x, y coefficient of correlation such as linear, curved, etc. This method is called "regression analysis".
This program permits 4 kinds of regression analysis.

P1: When data can be used in a linear regression formula
$y = a + bx$

P2: When you think the data can be used in an exponential regression formula
$y = a \cdot e^{bx}$

P3: When you think the data can be used in a logarithmic regression formula
$y = a + b \cdot \ln x$ (ln: natural logarithm)

P4: When you think the data can be used in a power regression formula
$y = a \cdot x^b$

PØ: Output. After all data input, a, b and r^2 are output and based on this, if you input x, the estimated value of y will be displayed.

Example

Analyze the relation between sales of ballpoint pens and those of pencils using the linear regression analysis.

In the case of linear regression

Item	1st day	2nd day	3rd day	4th day
Ballpoint pen (Xi)	14	19	26	34
Pencil (Yi)	29	22	16	9

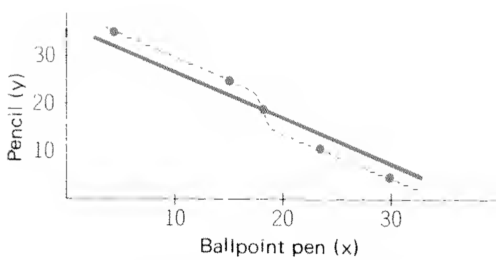
Operation

Step	Key operation	Display
	MODE Ø	
1	S P1	REGRESSION ANALYSIS -1 :
2	EXE	LINEAR
3	EXE	ROUND 1

Step	Key operation	Display
4	EXE	x= ?
5	14 EXE	y= ?
6	29 EXE	ROUND 2
7	EXE	x= ?
	Continue repeating this and input data.	
8	S P0	a=41.66041896
9	EXE	b=−0.9746416759
10	EXE	r ↑ 2=0.9880541759
11	EXE	x= ?
12	30 EXE	EXPECTED y= (Estimated value of y)
13	EXE	12.42116869

After completion of data input goes to P0

Therefore, the linear regression formula is as follows:
$y = 41.66 + (-0.974)x$



Since the correlation coefficient (r^2) is 0.988 ..... , these two items are much related.
It is predicted that 12.4 pencils will be sold when 30 ballpoint pens are sold.

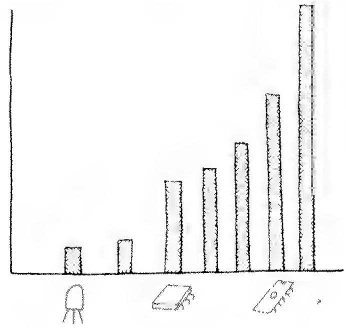
Reference

Category	Linear regression	Exponential regression	Logarithmic regression	Power regression
Program	P 1	P 2	P 3	P 4
Regression formula	$y = a + bx$	$y = a \cdot e^{bx}$	$y = a + b \cdot \ln x$	$y = a \cdot x^b$
X	x_i	x_i	$\ln x_i$	$\ln x_i$
Y	y_i	$\ln y_i$	y_i	$\ln y_i$
c	a	ln a	a	ln a
P	1	2	3	4

Annual average growth rate

<A.A.G.R.>

Any change in annual growth such as various achievements, prices, etc.



Preparation prior to program input

MODE 1 CLEARA EXE

S DEFM V 0 EXE

```
P0
  1 PRINT "A.A.G.R.
    "
  2 INPUT "1ST SALE
    S",A,"2ND SALES
    ",B,"DURATION:D
    ",D:SET F1
  3 E=LN (B/A)/D
  4 F=EXP 1
  5 R=F↑E*100-100
  6 PRINT "RATE=";R
    ;"%":SET N:GOTO
    2
```

Total 123 steps

Memory Contents

A	First sales
B	Second sales
D	Duration
E	Natural logarithm of average growth rate
R	Result of average growth rate

Explanation

Average growth and growth rate can be used in all required fields. However, at this time, let's take a look at a comparison of product sales.
Make the first category that you want to compare "1ST SALES" and make the second following that "2ND SALES".
Make the period covering this "DURATION : D", and obtain "RATE" (annual average growth rate).

Example

1st Sales	:	$500,000			
2nd Sales	:	$870,000			
Duration (years)	:	1 year	2 years	3 years	4 years
Growth Rate (%)	:	74.0	31.9	20.3	14.9

Operation

Step	Key operation	Display
	MODE 0	
1	S 0	A.A.G.R.
2	EXE	1ST SALES ?
3	500000 EXE	2ND SALES ?
4	870000 EXE	DURATION : D ?
5	1 EXE	RATE =74.0%
6	EXE	1ST SALES ?
7	500000 EXE	2ND SALES ?
8	870000 EXE	DURATION : D ?
9	2 EXE	RATE =31.9%

Repeated hereafter

Decimal ⟷ base-n conversion

<BASE CONVERTER>

Performs base conversion required in programming or controlling. Mutual conversion of decimal ⟷ base-n. Decimal numbers are OK too !

Preparation prior to program input

MODE 1 CLEARA EXE

S DEFM 0 EXE

```
P0
 1 PRINT "D TO A":GOSUB #9:GOSUB #7
 2 $="":V=30:INPUT "DECIMAL ",S:T=INT S:U=FRAC S
 3 Q=INT (T/R):X=T-Q*R:T=Q:V=V-1:$=A$(X)+$:IF T≠0 THEN 3
 4 IF U=0 THEN 8
 5 V=V-1:$=$+".":FOR W=V TO 1 STEP -1:U=U*R:X=INT U:U=FRAC U
 6 $=$+A$(X):IF U=0 THEN 8
 7 NEXT W
 8 PRINT S;"=",$;"(";R;")":GOTO 2
 9 NEXT W
10 PRINT S;"=",$;"(";R;")":GOTO 4
```

```
P1
 1 PRINT "A TO D":GOSUB #9:GOSUB #7
 2 S=0:Y=8:$=""
 3 INPUT "1-DG ",Z$
 4 IF Z$=".";Y=9:T=0:$=$+Z$:GOTO 3
 5 GOSUB #8:IF W=1 THEN 3
 6 $=$+Z$:GOSUB Y:PRINT $;"(";R;")=",S:GOTO 3
 8 S=S*R+X:RETURN
 9 T=T-1:S=S+X*R↑T:RETURN
```

```
P7
 1 INPUT "BASE ",R:R=INT ABS R:IF R*(R-1)=0 THEN 1
 2 IF R>16 THEN 1
 3 RETURN
```

```
P8
 1 FOR X=0 TO R-1:IF Z$=A$(X);W=0:RETURN
 2 NEXT X:PRINT Z$;":ERROR":W=1:RETURN
```

```
P9
 1 $="0123456789ABCDEF":FOR Q=1 TO 16:A$(Q-1)=MID(Q,1):NEXT Q
 2 RETURN
```

Total 532 steps

Memory Contents	
A ‧ P	Base-n number table
Q	Control variable, Quotient
R	BASE
S	Decimal
T	Integer portion of S
U	Decimal portion of S
V $	Character number counter
W	Control variable, (Error, flag)
X	Remainder
Y	Indirect GOTO pointer
Z	Character readout buffer

Explanation Performs conversion between decimal and another base. Generally, binary and hexadecimal are representative. Conversion can be made to hexadecimal. Decimal numbers are also possible.

P0 : D TO A

P1 : A TO D

D : Decimal

A : Another base ($n \leqq 16$)

Since the BASE is asked, make input accordingly. Thereafter, input according to the display.

Example What is the hexadecimal for a decimal of 58.5? What is the decimal for a hexadecimal of 35B?

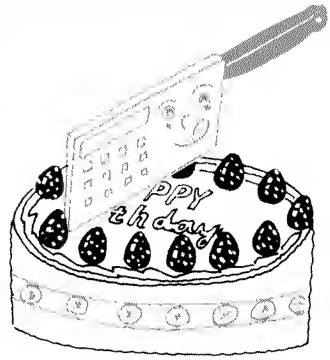
Operation

	Step	Key operation	Display
		MODE 0	
D TO A ⇒ P0	1	S P0	D TO A
	2	EXE	BASE ?
	3	16 EXE	DECIMAL ?
	4	58.5 EXE	58.5 =
	5	EXE	3A. 8 (16)
A TO D ⇒ P1	6	S P1	A TO D
	7	EXE	BASE ?
	8	16 EXE	1- DG ?
Input from the most significant digit. ⇒	9	3 EXE	3 (16)=
	10	EXE	3 1-DG? at the next EXE
	11	5 EXE	35 (16)=
	12	EXE	53 1-DG? at the next EXE
	13	B EXE	35B (16)=
	14	EXE	859

Least common multiple and greatest common measure

<LCM, GCM>

Both are possible using one program called a "greedy" program.



Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [V] (DEFM) 0 [EXE]

```
P0
10 PRINT "LCM,GCM"
11 P=200:$="LCM"
12 INPUT "KEY IN L
   OR G ",L$
13 IF L$="L" THEN
   20
14 IF L$≠"G" THEN
   12
15 P=202:$="GCM"
20 INPUT "A=",A:IN
   PUT "B=",B:IF A
   *B=0 THEN 20
30 C=A:D=B:IF C≥D
   THEN 50
40 E=D:D=C:C=E
50 E=C
60 IF C≠D THEN 80
70 PRINT $;"=";E:G
   OTO 20
80 GOSUB P
90 F=2:GOSUB 120:F
   =3
100 GOSUB 120:IF F>
    INT (SQR (D*F)/
    2)*2+1 THEN 70
110 F=F+2:GOTO 100
120 X=C:GOSUB 150:I
    F R≠0;RETURN
130 X=D:GOSUB 150:I
    F R≠0;RETURN
140 GOSUB P+1:C=C/F
    :D=D/F:GOTO 120
150 R=X-INT (X/F)*F
    :RETURN
200 E=C*D:RETURN
201 E=E/F:RETURN
202 E=1:RETURN
203 E=E*F:RETURN
```

Total 362 steps

Memory Contents	
F	Temporary LCM or GCM
P	Indirect address pointer for LCM and GCM switching

Explanation This program combines least common multiple (LCM) and greatest common measure (GCM). The differences between the two processes are made into subroutines. The respective A values and B values are obtained.

Operation

LCM

Step	Key operation	Display
	MODE 0	
1	S (0)	LCM, GCM
2	EXE	KEY IN L OR G ?
3	(LCM) L EXE	A= ?
4	(Ex.) 35 EXE	B= ?
5	(Ex.) 7 EXE	LCM= 35

GCM

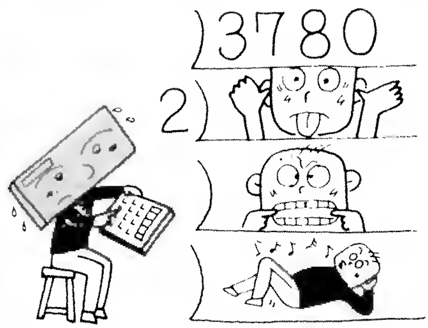
Step	Key operation	Display
	MODE 0	
1	S (0)	LCM, GCM
2	EXE	KEY IN L OR G ?
3	(GCM) G EXE	A= ?
4	(Ex.) 35 EXE	B= ?
5	(Ex.) 7 EXE	GCM= 7

If the numbers become greater, the calculation will take time. Sometimes you may have to wait anywhere from a few seconds to a few minutes to get the answer.
However, even so, the PC-4 performs the calculations quickly.

Prime factorization

<PRIME-I>

Prime factorization is effective in simplifying complicated calculations. Mathematics can be enjoyable.!



Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 0 [EXE]

```
P0
  10 PRINT "PRIME-I"
  20 INPUT "A=",Z:A=
     Z:IF A=1 THEN 2
     0
  30 I=2:GOSUB 50:I=
     3:GOTO P
  40 GOSUB 50:I=I+2:
     GOTO P
  50 M=0:B=INT ((SQR
      A)/2)*2+1:IF I
     >B THEN 100
  60 S=SGN (A-INT (A
     /I)*I)-1:M=M-S:
     A=A*I↑S:GOTO (7
     0+S*10)
  70 IF M≠0 THEN 120
  80 P=40:IF A=1;P=2
     0
  90 RETURN
 100 IF A=Z THEN 130
 110 I=A:M=1:A=1
 120 PRINT "N=";I;"
     ↑";M:PRINT "QUO
     =";A:GOTO 80
 130 PRINT Z;" IS PR
     IME NO.":P=20:R
     ETURN
```

Total 258 steps

Memory Contents	
A	Z / I when divisible
B	Upper limit of I
I	Temporary prime factor
M	Exponent of same prime factor
P	Pointer for the destination after returning from subroutine 50.
Z	Input data

Explanation For prime factorization for some number, you must find a prime number which has no exact divisor other than 1 and the number itself, such as 2, 3, 5, 7, 11, etc.

$$a = 2^a \times 3^b \times 5^c \times 7^d \cdots\cdots$$

In the program, input data A is divided by temporary prime factor I and if divisible, A will be updated using A/I and M will be increased by 1. This is repeated until no longer divisible and I↑M and the quotient are displayed, then I is updated and the same operation is repeated.
If I exceeds $\sqrt{A}$, Z is considered to be the prime number.

Operation

N: prime factor
QUO: quotient

Step	Key operation	Display
	MODE 0	
1	S P0	PRIME-I
2	EXE	A=?
3	(Ex.) 385 EXE	N=5↑1
4	EXE	QUO=77
5	EXE	N=7↑1
6	EXE	QUO=11
7	EXE	N=11↑1
8	EXE	QUO=1

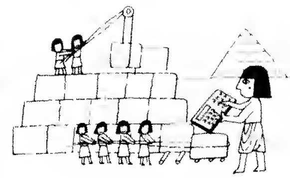
After the above, the following answer will be displayed.

$$385 = 5^1 \times 7^1 \times 11^1$$

Universal power calculation

<UNIVERSAL P-CAL>

Can be used for square root, cube root, square and cube as well as other power calculations.



Preparation prior to program input

MODE 1 CLEARA EXE

S DEFM 0 EXE

```
P0
  10 PRINT "SQUARE R
     OOT-CAL":E=.5
  20 GOSUB #6:PRINT
     "SQUARE ROOT(";
     X;")=",Y:GOTO 2
     0
P1
  10 PRINT "CUBE ROO
     T-CAL":E=1/3
  20 GOSUB #6:PRINT
     "CUBE ROOT(";X;
     ")=",Y:GOTO 20
P2
  10 PRINT "SQURE-CA
     L":E=2
  20 GOSUB #5:GOTO 2
     0
P3
  10 PRINT "CUBE-CAL
     ":E=3
  20 GOSUB #5:GOTO 2
     0
P4
  10 PRINT "UNIV-P-C
     AL":INPUT "EXP=
     ",E
  20 GOSUB #5:GOTO 2
     0

P5
  30 GOSUB #6:PRINT
     X;" ↑";E;"=";Y:
     RETURN
P6
  10 INPUT "X=",X:IF
      X≤0 THEN 10
  20 Y=X↑E:RETURN
```

Total 265 steps

Memory Contents	
E	Exponent
X	Input data
Y	Answer: Y = X ↑ E

Explanation Using $y = x^E$, most frequently used items such as square root, cube root, square and cube can be obtained independently. With respect to these, since exponent E is given in the program, by simply inputting X, Y will be output. Using PØ and P1, the output will be in the form SQUARE ROOT (X) = answer or CUBE ROOT (X) = answer.

- PØ : Square root $y = \sqrt{x}$
- P1 : Cube root $y = \sqrt[3]{x}$
- P2 : Square $y = x^2$
- P3 : Cube $y = x^3$
- P4 : Other power calculations. First, input E. Next, if X is input, the answer will be given as Y = X↑E. All input is made following the "?".

Operation

Square root PØ

Step	Key operation	Display
	MODE Ø	
1	S PØ	SQUARE ROOT-CAL
2	EXE	x = ?
3	3 EXE	SQUARE ROOT (3)=
4	EXE	1.732050808

The answers of P1, P2 and P3 are obtained in the same manner as above.

Simultaneous linear equations

By using BASIC, even this level of mathematics can be enjoyable !

Simultaneous linear equations with 2 or 3 unknowns

2 unknowns — P2

$$\begin{cases} A_1x_1 + B_1x_2 = C_1 \\ A_2x_1 + B_2x_2 = C_2 \end{cases}$$

3 unknowns — P3

$$\begin{cases} A_1x_1 + B_1x_2 + C_1x_3 = D_1 \\ A_2x_1 + B_2x_2 + C_2x_3 = D_2 \\ A_3x_1 + B_3x_2 + C_3x_3 = D_3 \end{cases}$$

Preparation prior to program input

[MODE] [1] CLEARA [EXE]

[S] [DEFM V] 0 [EXE]

```
P1
  1 PRINT "S.L.EQ";
    -V
  2 U=0
  3 GOSUB #9:FOR W=
    1 TO V:F(W)=A(W
    ):NEXT W:X=2:GO
    SUB 9
  4 FOR W=2 TO V:J(
    W)=A(W):NEXT W:
    IF V=2;B=H-G*C:
    GOTO 7
  5 X=3:GOSUB 9:FOR
    W=3 TO V:M(W)=A
    (W):NEXT W
  6 C=M-L*D:B=I-G*C
    -H*D
  7 FOR W=1 TO V:PR
    INT "X";W;" = "
    ;A(W):NEXT W:GO
    TO 2
  8 FOR Y=2 TO X:GO
    SUB 10*Y:NEXT Y
    :RETURN
  9 GOSUB #9:GOSUB
    8:RETURN
 20 IF A=0 THEN 22
 21 FOR W=1 TO V:A(
    W)=A(W)-F(W):NE
    XT W
 22 IF B=0 THEN 24
 23 FOR W=2 TO V:A(
    W)=A(W)/B:NEXT
    W:RETURN
 24 IF U=2;GOSUB #8
    :GOTO 20
 25 RETURN
 30 IF B=0 THEN 32
 31 FOR W=2 TO V:A(
    W)=A(W)-J(W):NE
    XT W
 32 IF C=0 THEN 34
 33 D=D/C:RETURN
 34 IF U=3;GOSUB #8
    :GOTO 30
 35 RETURN
P2
  1 V=2:GOTO #1
P3
  1 V=3:GOTO #1
P8
  1 PRINT "PIVOT=0"
    ,"RETRY!":U=U-1
    :RETURN
P9
  1 U=U+1
  2 PRINT "ROUND";U
    :INPUT "A= ",A:
    IF A+U=1;GOSUB
    #8:GOTO 2
  3 INPUT "B= ",B,"
    C= ",C:IF V=2 T
    HEN 7
  4 INPUT "D= ",D
  7 IF A=0;RETURN
  8 FOR W=1 TO V:A(
    W)=A(W)/A:NEXT
    W:RETURN
```

1 PIVOT: Changes the sequence of ROUND 1 and ROUND 2.

2 RETRY!: Try again!

Total 541 steps

Memory Contents			
A	Coefficients and constant terms	M	D of ROUND 2
~		P	D of ROUND 3
D		U	Counter for number of ROUNDs
G	B through D of ROUND 1	V	Number of unknowns
H		W	Control variables
I		X	
L	C of ROUND 2	Y	Pointer for indirect GOTO statement

Explanation This is a program which performs simultaneous linear equations both with 2 and 3 unknowns.

P2: Simultaneous linear equation with 2 unknowns.

$$\begin{cases} A_1x_1 + B_1x_2 = C_1 & \text{.............. (ROUND 1)} \\ \\ A_2x_1 + B_2x_2 = C_2 & \text{.............. (ROUND 2)} \end{cases}$$

First, the PC-4 asks the values A, B and C of ROUND 1.
Next, it asks the values, A, B and C of ROUND 2. After these are input, the results (x_1, x_2) will be displayed.

P3: Simultaneous linear equation with 3 unknowns. In the same manner as above, the results (x_1, x_2, x_3) will be displayed after the values A through D of ROUND 1, 2 and 3 are input.

Example

$$\begin{cases} 5x_1 + 2x_2 + 4.5x_3 = 68 \\ 4x_1 + 3.6x_2 + 7x_3 = 80.8 \\ 9x_1 + 7x_2 + 8x_3 = 132 \end{cases}$$

Obtain x_1, x_2 and x_3.

Operation

Step	Key operation	Display
	MODE 0	
1	S 3	S.L.EQ—3
2	EXE	ROUND 1
3	EXE	A= ?
4	5 EXE	B= ?
5	2 EXE	C= ?
6	4.5 EXE	D= ?
7	68 EXE	ROUND 2
8	EXE Hereafter, input the values up to D of ROUND 3 by repeating steps 2 through 7.	A= ?
9 (Result of x_1)	EXE	X1=7
10 (Result of x_2)	EXE	X2=3
11 (Result of x_3)	EXE	X3=6
12	EXE	ROUND 1 (Returns to step 2.)

Simultaneous linear equation with 4 unknowns can also be programmed within 544 steps. With reference to this program, try it!

Function List

Function Name	Use Example	Function	Argument Range
SIN	SIN x	Sine function of x	DEG $\lvert x\rvert < 1440$
COS	COS x	Cosine function of x	RAD $\lvert x\rvert < 8\pi$
TAN	TAN x	Tangent function of x	GRA $\lvert x\rvert < 1600$
ASN	ASN x	Arcsine function of x	$\lvert x\rvert \leqq 1$
ACS	ACS x	Arccosine function of x	
ATN	ATN x	Arctangent function of x	Entire range
LOG	LOG x	Common logarithm of x	$x > 0$
LN	LN x	Natural logarithm of x	$x > 0$
EXP	EXP 1	Function to call out the numerical value of the exponential table.	
SQR	SQR x	Square root of x	$x \geqq 0$
INT	INT x	Maximum integer which does not exceed x	Entire range
FRAC	FRAC x	Decimal portion of x	Entire range
ABS	ABS x	Absolute value of x	Entire range
SGN	SGN x	Sign of x $\begin{pmatrix} x<0\rightarrow -1, & x=0\rightarrow 0 \\ x>0\rightarrow 1 & \end{pmatrix}$	Entire range
RAN#	RAN#	Random number greater than 0 but less than 1	
RND	RND(x, y)	Value of x rounded off at the $10^{y'}$ position, provided that y' is a value where y is rounded off below the decimal point	x : Entire range y : $-100 < y < 100$
LEN	LEN(A$)	Uses a numerical value to determine the length of the character string which is stored in the character variable or $ memory	Simple character variable (A$~Z$) and $ memory
VAL	VAL(A$)	Converts the character string which is stored in the character variable or $ memory into a numerical value (Example) A$="123"→VAL(A$)=123	〃
MID	MID(x) MID(x, y)	Extracts y characters from the character string stored in the $ memory starting from the xth character position If y is omitted, extracts all of the characters starting from the xth position	$1 \leqq x,\ y < 31$ $x+y-1 \leqq 30$
KEY	KEY	The key which is pressed during program execution is read in as a character	

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO, L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	UK
260-316 VICTORIA ROAD RYDALMERE N S W 2116	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

12A2A

㊕ Printed in Japan